Universidad Carlos III de Madrid

Alexandru Bikfalvi

Peer-to-Peer Television for the IP Multimedia Subsystem

Universidad Carlos III de Madrid

Peer-to-Peer Television for the IP Multimedia Subsystem

Author
Alexandru Bikfalvi

Advisor
Dr. Jaime García-Reinoso

2012

Peer-to-Peer Television for the IP Multimedia Subsystem

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Prepared by
Alexandru Bikfalvi

Under the advice of
Dr. Jaime García-Reinoso

Departamento de Ingeniería Telemática
Universidad Carlos III de Madrid

Date: July 2012

Web: http://alex.bikfalvi.com/research/phd

Credits: This work has been supported by the Madrid Institute for Advanced Studies in Networks (Institute IMDEA Networks) and by a FPU scholarship from the Spanish Ministry of Education.

The photgraphs from the front cover are authored by Aurelio Asiain, Mark Barnes, Richard Bartz, Böhringer Friedrich, Sam Garza, Georgios Giannopoulos, Benjamin Gimmel, David Gubler, Kim Hansen, Mikey Hennessy, Brocken Inaglory, Shalom Jacobovitz, Raul Kern, Holger Krisp, Greg O'Beirne, Simo Räsänen, Pierre Selim, Silvio Tanaka, Berthold Werner.

Thank you

To my doctoral advisor, Jaime García-Reinoso, to my family, to my friends from the Madrid Institute for Advanced Studies in Networks, the Telematics Engineering Department and the Robotics Lab of Universidad Carlos III de Madrid, the NeTS group of Universitat Pompeu Fabra, and to my fellow musicians from Sambanés. For the support, encouragement, and great moments we spent together these years, I am to all of you extremely grateful.

Resumen

La transmisión de vídeo con tecnologías peer-to-peer (P2P) ha generado un gran interés, tanto en la industria como en la comunidad científica, quienes han encontrado en dicha unión la solución para afrontar los problemas de escalabilidad de la transmisión de vídeo, reduciendo al mismo tiempo sus costes. A pesar del éxito de estos mecanismos en Internet, la transmisión de vídeo mediante técnicas P2P no se ha utilizado en servicios comerciales como puede ser el de televisión por IP (IPTV). Con la aparición de propuestas de redes de próxima generación basadas en el IP Multimedia Subsystem (IMS), que permite una arquitectura abierta e interoperable, los mecanismos basados en P2P emergen como posibles alternativas en situaciones donde los mecanismos tradicionales de transmisión de vídeo no se pueden desplegar o no son económicamente viables.

Esta tesis propone una arquitectura de servicio de televisión peer-to-peer para una red de siguiente generación basada en IMS, que abreviaremos como P2PTV, que permite a uno o más proveedores de servicio utilizar una infraestructura P2P común para la transmisión de canales de TV a sus suscriptores. En vez de utilizar varios servidores, proponemos utilizar la capacidad de envío de los equipos de usuario, como los set-top boxes, localizados en el lado del cliente. En esta tesis extendemos los trabajos de estandarización sobre IMS IPTV de los organismos 3rd Generation Partnership Project (3GPP) y del Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN), donde un servidor de aplicación (AS) central de P2PTV administra el acceso de los clientes al servicio y permite compartir los recursos de los equipos.

Debido a que el acceso a los canales de TV por parte de los usuarios es una actividad compleja, nos enfrentamos a dos retos importantes. El primero es administrar la señalización de IMS, con la cual se reservan los recursos de QoS necesarios durante cada cambio de canal, estableciendo una sesión multimedia entre los diferentes elementos de la comunicación. El segundo está representado por las interrupciones de la reproducción de video, causado por los equipos que sirven dicho vídeo cuando estos se desconectan del sistema o cuando cambian de canal.

Para afrontar estos retos, proponemos dos mejoras al sistema. La primera mejora introduce el método de *señalización rápida*, en la cual se utilizan sesiones multimedia inactivas pero con recursos reservados para acelerar las conexiones entre usuarios. En cada momento, el AS utiliza la información extraída del algoritmo propuesto, que calcula el número de sesiones necesarias para administrar la demanda de conexiones, pero sin realizar una sobre-estimación, manteniendo bajo el uso de los recursos. Hemos abordado con especial cuidado la movilidad de los usuarios, donde se ha propuesto una transferencia de sesión pro-activa utilizando el estándar IEEE 802.21, el cual brinda una mejor alternativa que los métodos propuestos hasta la fecha.

La segunda mejora se enfoca en las *desconexiones de usuarios* durante cambios de canal. Dividiendo los canales de TV en varios segmentos, permitimos a los equipos descargar o enviar diferentes partes de cualquier canal, aumentando la estabilidad de su participación. A diferencia de otros trabajos, nuestra propuesta se beneficia de la estimación de la demanda futura de los usuarios, proponiendo un método descentralizado para una asignación balanceada del ancho de banda de los equipos.

Hemos evaluado el rendimiento del sistema P2PTV a través de modelado y de simulaciones de ordenador en sistemas IPTV de gran escala. Una configuración simple, con envío P2P puro, indica mejoras en el retardo y número de desconexiones de usuarios. En escenarios más complejos, especialmente con equipos con pocos recursos en la subida, sugerimos el uso de P2P como una solución complementaria a las soluciones tradicionales de multicast IP. Reservando el uso de P2P para los canales de TV poco populares, se permite explotar los recursos de los equipos y se previene la necesidad de un alto número de árboles multicast dispersos. Como trabajo futuro, se propone refinar los algoritmos del AS, abordar diferentes escenarios experimentales y también extender las lecciones aprendidas en esta tesis a otros sistemas no basados en IMS.

Abstract

Peer-to-peer (P2P) video streaming has generated a significant amount of interest in both the research community and the industry, which find it a cost-effective solution to the user scalability problem. However, despite the success of Internet-based applications, the adoption has been limited for commercial services, such as Internet Protocol Television (IPTV). With the advent of the next-generation-networks (NGN) based on the IP Multimedia Subsystem (IMS), advocating for an open and inter-operable architecture, P2P emerges as a possible alternative in situations where the traditional mechanisms are not possible or economically feasible.

This work proposes a P2P IPTV architecture for an IMS-based NGN, called P2PTV, which allows one or more service providers to use a common P2P infrastructure for streaming the TV channels to their subscribers. Instead of using servers, we rely on the uploading capabilities of the user equipments, like set-top boxes, located at the customers' premise. We comply with the existing IMS and IPTV standards from the 3rd Generation Partnership Project (3GPP) and the Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN) bodies, where a centralized P2PTV application server (AS) manages the customer access to the service and the peer participation.

Because watching TV is a complex and demanding user activity, we face two significant challenges. The first is to accommodate the mandatory IMS signaling, which reserves in the network the necessary QoS resources during every channel change, establishing a multimedia session between communicating peers. The second is represented by the streaming interruptions, or churn, when the uploading peer turns off or changes its current TV channel.

To tackle these problems, we propose two enhancements. A *fast signaling* method, which uses inactive uploading sessions with reserved but unused QoS, to improve the tuning delay for new channel users. At every moment, the AS uses a feedback based algorithm to compute the number of necessary sessions that accommodates well the demand, while preventing the over-reservation of resources. We approach with special care mobility situations, where a proactive transfer of the multimedia session context using the IEEE 802.21 standard offers the best alternative to current methods.

The second enhancement addresses the *peer churn* during channel changes. With every TV channel divided into a number of streams, we enable peers to download and upload streams different from their current channel, increasing the stability of their participation. Unlike similar work, we benefit from our estimation of the user demand and propose a decentralized method for

a balanced assignment of peer bandwidth.

We evaluate the performance of the P2PTV through modeling and large-scale computer simulations. A simpler experimental setting, with pure P2P streaming, indicates the improvements over the delay and peer churn. In more complex scenarios, especially with resource-poor peers having a limited upload capacity, we envision P2P as a complementary solution to traditional approaches like IP multicast. Reserving P2P for unpopular TV channels exploits the peer capacity and prevents the necessity of a large number of sparsely used multicast trees. Future work may refine the AS algorithms, address different experimental scenarios, and extend the lessons learned to non-IMS networks.

Contents

List of Figures

List of Tables

Part

I

Background

Chapter 1

Introduction

1.1 Television in the IP Multimedia Subsystem

Although the idea of television over the Internet Protocol (IP), or IPTV, has been existing for many years, only recently it has gained a significant attention from the service providers. This interest has led to several commercial deployments, usually bundled with telephony and Internet access service packages, commonly referred to as triple play. The catalyst for these changes stems from the economic advantage of providing both high-speed and traditional services, like voice, over the same broadband connection. However, as the broadband access is being transformed into a commodity, fewer subscribers choose to contract these classic options, and instead reorient themselves toward the cheaper market of Internet services. In these cases, the telcos can lose their infrastructure investments, as the bulk of money flow goes directly to the third party providers.

Because services such as VoIP and IPTV are gradually replacing their legacy counterparts, both the industry and the academia have directed a lot of effort into designing and standardizing a *next-generation network*, or NGN, that will allow both the telcos and third party providers to compete in the service market. The next-generation network emerged as an integrated broadband network that could act as a flexible platform for any type of service, by decoupling the service provisioning from the transport network through a set of standardized interfaces (Lee & Knight, 2005). In addition to the business incentives, the next-generation networks are designed to guarantee from the start an adequate level of quality of service (QoS). Toward this end, every service may reserve and commit its necessary network resources, such as bandwidth and latency requirements.

The most widely known next-generation network implementation is based on the IP Multimedia Subsystem, or IMS (3GPP, 2012c). Standardized by the Third Generation Partnership Project (3GPP), the IMS is an architectural framework to deliver IP multimedia services, where the Session Initiation Protocol, or SIP (Schulzrinne et al., 2003), is the main session control protocol. Initially developed as the next-generation evolution to the current cellular mobile telephony, the Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN) working group from the European Telecommunications Standards Institute (ETSI) is

- Convergence of multimedia services
- Guaranteed quality-of-service
- Session signaling using SIP

- Channel demand estimation
- Fast session setup
- Reduced peer churn

Arguments against IP multicast

- Large number of TV channels: many providers, near video-on-demand, user generated content
- Static multicast: inefficient for many TV channels
- Dynamic multicast: delay and scalability issues
- Administrative and economic reasons
- Support for multiple transport protocols

Figure 1.1: Arguments against IP multicast, as singular solution for IPTV streaming.

working to extend its functionality for fixed transport technologies. In this context, the next-generation network is the result of various integrated networks under the umbrella of the IMS, and where multimedia services are seamlessly available to the users.

Among these, the broadcast television represented an integral part of the ongoing TISPAN standardization effort (TISPAN, 2011a). This recognizes the key role that television will play in the near future, despite the evolution of the communication technologies and the increasing level of interactivity. Their solution proposes IP multicast (Deering, 1986) for the streaming of the audio/video content of the available TV channels, having the advantage of high performance and availability with existing protocols and equipments. This approach mirrors the current commercial-grade IPTV deployments, in which telcos use purposefully designed network infrastructures and have complete control over the allowed traffic. Commonly referred to as walled-gardens, these architectures rely on set-top boxes at the customers premise to provide a traditional TV viewing experience.

Despite these success stories, we argue the IP multicast may not always be the best technological choice. With the advent of the next-generation networks, which advocate for an open and inter-operable infrastructure, we may reasonably expected that the future networks have to support a larger number of service providers, and consequently a large number of TV channels. Semi-interactive techniques such as near video-on-demand (NVoD), where the same TV program is broadcast several times, will increase the number of channels as well. As a final argument, in today's Internet we witness a large growth of user generated video content, where most of it is published as content-on-demand. However, similar to the demand for live audio streaming, that spawned a large number of Internet radio channels, we argue that even residential users can be interested in generating their own live TV content.

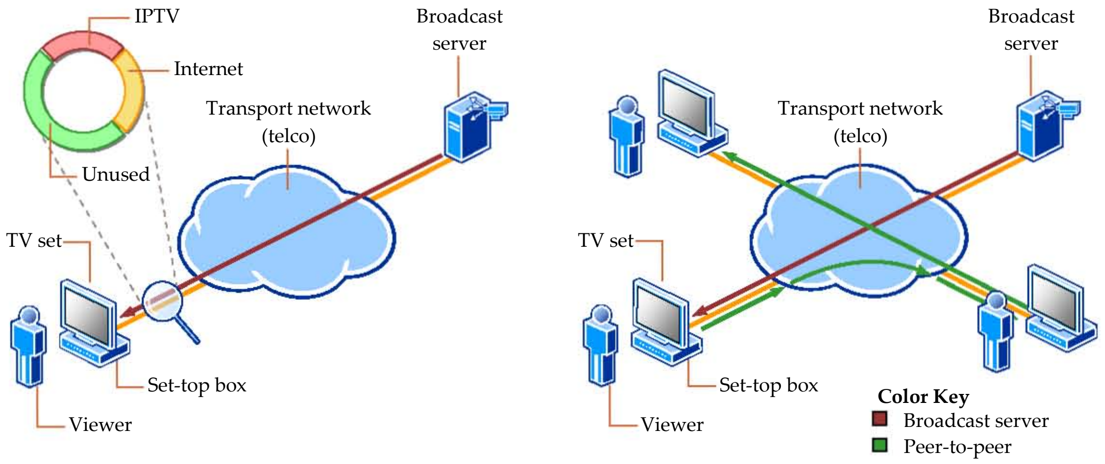
These trends inherently expose the network operators to a number of issues, as follows and summarized in the figure 1.1.

- Under these circumstances it is no longer affordable to use static IP multicast to stream all TV channels, as some IPTV providers do at the present.

- Dynamic multicast can lead to scalability problems that have been studied extensively by the research community (Wong & Katz, 2000; Thaler & Handley, 2000; Fei et al., 2001; B. Zhang & Mouftah, 2003). These can translate to an increased performance cost for the service provider, especially when the number of users viewing a multicast channel is low.

- The problem is aggravated through the use of scalable video coding techniques, such as H.264/SVC, which would require multiple multicast trees for every TV channel (ITU-T, 2011).

- Multicast group management, including authorization to create new multicast groups, global scope multicast address allocation, sender and/or receiver authorization, represent a set of complex issues to solve in a multi-domain scenario.



(a) Classic IPTV streaming.

(b) Peer-to-peer IPTV streaming.

Figure 1.2: The concept behind peer-to-peer television.

- The Transmission Control Protocol (TCP) does not work well with IP multicast. Nowadays much of the multimedia traffic works over UDP but there are some initiatives to use a different approach, like using the Datagram Congestion Control Protocol (DCCP).

- Even when the previous technical challenges are mitigated, something feasible in a closed, telco-owned network, in the situation of a next-generation network, which requires interoperability with a number of external providers, multicast could still be defeated by policy issues.

1.2 An Overview of Our Work

By carefully considering these trends, in this dissertation, we propose a *peer-to-peer television service for the IP Multimedia Subsystem*, or P2PTV, exploring the use peer-to-peer (P2P) as an alternative to the traditional IP multicast streaming. The core principle behind this approach, as illustrated in the figure 1.2(a), is to exploit the unused download and upload capacity from the network edges, such as bandwidth which is physically available in the network but not contracted and paid for by the subscriber.

In the peer-to-peer scenario, shown in the figure 1.2(b), the user equipments (UE), often referred to as *peers*, upload some of the receiving live streams to the equipment of other subscribers requesting the same TV channel. Given the setting of our work, we take advantage of dedicated user equipments, such the household set-top box, which will perform the peer-to-peer streaming functions completely behind the scenes. The chief advantage of peer-to-peer is the inherent scalability with the number of P2PTV subscribers. The cost is the higher bandwidth usage and increased complexity, when compared with alternative mechanisms. We note however, that we only exploit network bandwidth that otherwise would remain unused.

To accommodate our proposal with the requirements of the IMS platform, a centralized P2PTV application server, or P2PTV-AS, manages the access to the service and coordinates the peering participation of the user equipments. A functional entity of the IMS, the P2PTV-AS performs its role by intermediating the establishment of streaming sessions. In this capacity, it is crucial for the selection of the uploading peers, or of the broadcast server when no other peers are available. The figure 1.3 sketches a high-level view of the P2PTV service architecture, emphasizing the design objectives for the UE and the P2PTV-AS.

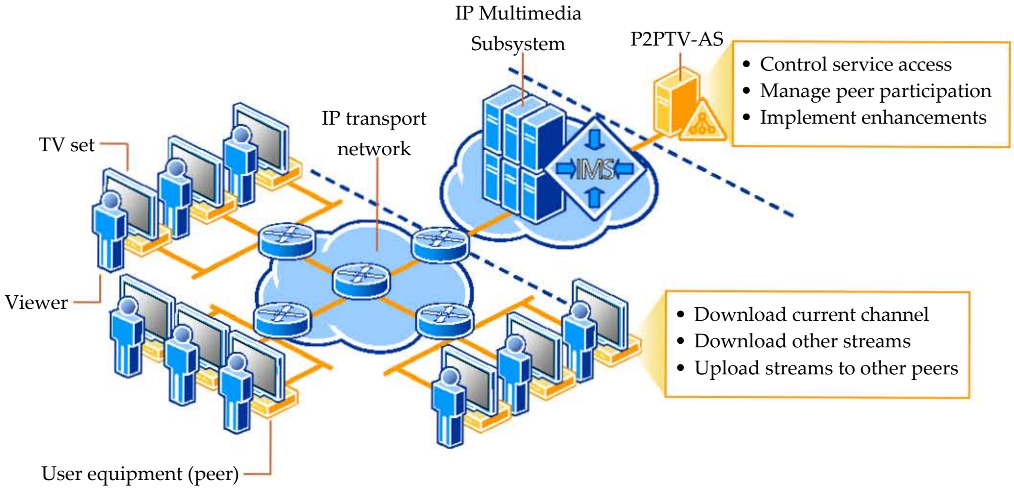


Figure 1.3: The high-level architecture of the P2PTV service.

Although peer-to-peer audio/video streaming is not new, with academic research and industry applications spanning more than a decade, we face the unique challenges of integrating peer-to-peer with the requirements of a commercial-grade IPTV service within the IMS. Whereas most previous work on peer-to-peer video, as an Internet application, focused on the delivery of singular streams, watching TV is a complex and demanding user activity, and supporting multiple TV channels raises a number of difficulties (Cha, Rodriguez, Crowcroft, et al., 2008; Qiu, Ge, Lee, Wang, Xu, & Zhao, 2009). Cha et al. found that over 60 % of channel changes occur within 10 seconds of zapping to a new channel. This activity pattern indicates the viewer browsing or surfing in the search of an interesting TV channel, and it is a reference to the expected operating conditions.

In addition, the advantages of an IMS-based next-generation network, such as service integration, authentication, charging and quality of service, come with the cost of a time-expensive SIP signaling exchange between the communicating parties during the setup or termination of any TV session. This situation is further worsened by the highly dynamic streaming environment where every channel change is manifested as a streaming interruption, or *churn*, affecting the overall performance.

To tackle these problems, we propose two enhancements.

- The first is a *fast signaling* method, designed to reduce the delay during a channel change. Toward this end, we propose using inactive uploading sessions, which are established SIP dialogs with reserved network resources. They reduce the number of exchanged SIP messages, as well as having the QoS resources already committed, requiring little further interaction with the network equipments.

- The second is a *low churn* method, with the purpose of avoiding uploading interruptions. Our algorithms, designed on the foundation laid by Wu et al. (D. Wu, Liang, et al., 2009), consist of a bandwidth assignment strategy which enables peer to participate on multiple TV channels at the same time. In turn, this provides a more stable connectivity between user equipments, less susceptible to disruptions generated by channel changes.

Both enhancements rely on a feedback control algorithm, implemented at the P2PTV-AS, which provides an estimation of the future number of viewers on every TV channel. This allows us to determine the best number of inactive sessions and the peer bandwidth assignment that will accommodate the service demand, while preventing the over-reservation of resources. We pay special attention to mobility scenarios, by comparing different solutions based on SIP and Mobile

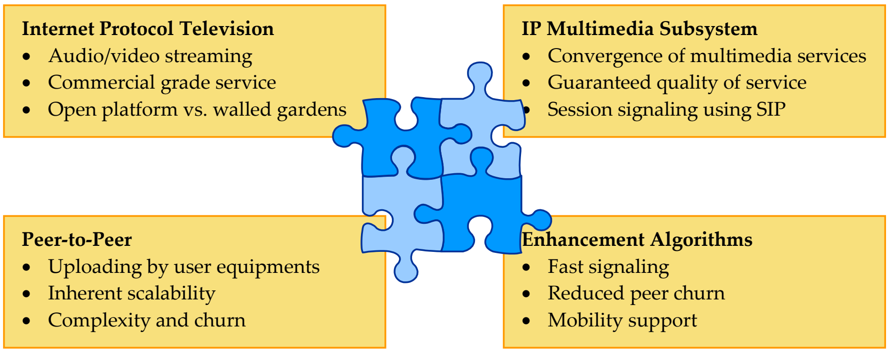


Figure 1.4: The main topics covered by the present work.

IP (MIP). Considering the high impact of the session setup delay on the quality of experience, we propose a mobility technique based on the proactive transfer of the multimedia session context between the old and new networks, using the IEEE 802.21 standard.

The figure 1.4 summarizes the main four topics that are part of this dissertation. To the best of our knowledge, this is one of the first proposals in this direction, addressing the IPTV, IMS and P2P integration from the perspective of a performance-oriented design.

Arguments against IP multicast

- Large number of TV channels: many providers, near video-on-demand, user generated content
- Static multicast: inefficient for many TV channels
- Dynamic multicast: delay and scalability issues
- Administrative, economic and policy issues reasons
- Support for multiple transport protocols

1.3 Summary of Contributions

The present dissertation is the result of a research effort that spanned several years. Throughout this period, it was our intention to ensure that the work has been recognized and accepted by the scientific community through a number of peer-reviewed publications. Considering the wide range of issues that have been addressed, each novel idea spans one or more published articles, as shown in the figure 1.5, whereas the figure 1.6 recollects briefly the corresponding history of our work. To this end, the main contributions are the following.

- First, we design a P2PTV service for the IP Multimedia Subsystem, which relies on the upload capability of the household user equipments to service requests for the live TV channel content. The architecture identifies the application server, or P2PTV-AS, as the main functional entity responsible for the peer-to-peer communication. We propose a business model where multiple IPTV service providers use the P2PTV service as a platform for live TV channel distribution (Bikfalvi et al., 2009a, 2009b).

- Given the constraints imposed by the IMS specifications to any multimedia session, the TV channels are divided into a constant number of streams or layers. To stream each layer, we resort to a push-pull mechanism along a application-level multicast tree. This approach offers the advantage of an increased granularity of peer participation, is more robust to peer churn, and it can be used with scalable video codecs (Vidal et al., 2009).

- Third, we add support for mobility scenarios. Through a brief analysis of existing mobility capabilities based on both SIP and Mobile IP, we design a proactive approach which transfers the context of the multimedia sessions before the mobile equipment changes the network. Our work uses the IEEE 802.21 Media Independent Handover standard to identify the roaming situation, detect the new configuration, and copy the session information (Vidal et al., 2010).

- The design of the performance-oriented enhancement algorithms requires a thorough understanding of the viewer activity in a large-scale IPTV system. Although there are many studies describing the user behavior in Internet-based streaming applications, our understanding

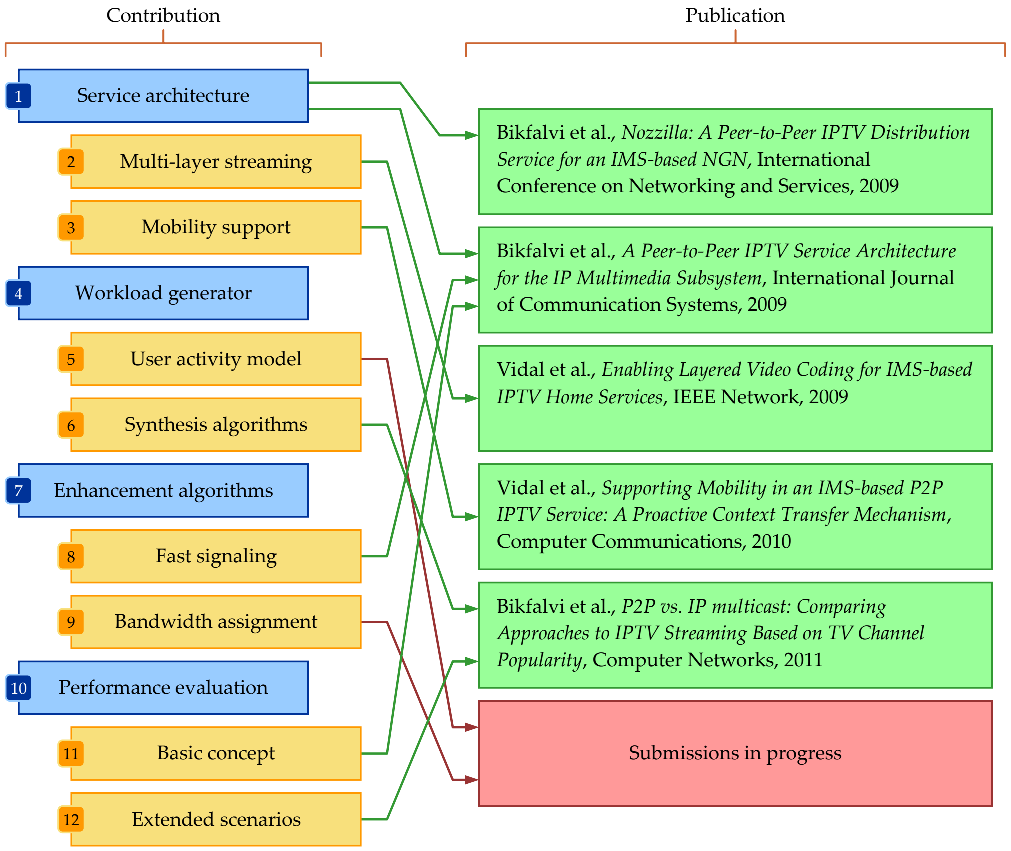

Contribution
Publication
1 Service architecture
2 Multi-layer streaming
3 Mobility support
4 Workload generator
5 User activity model
6 Synthesis algorithms
7 Enhancement algorithms
8 Fast signaling
9 Bandwidth assignment
10 Performance evaluation
11 Basic concept
12 Extended scenarios
Bikfalvi et al., *Nozzilla: A Peer-to-Peer IPTV Distribution Service for an IMS-based NGN*, International Conference on Networking and Services, 2009
Bikfalvi et al., *A Peer-to-Peer IPTV Service Architecture for the IP Multimedia Subsystem*, International Journal of Communication Systems, 2009
Vidal et al., *Enabling Layered Video Coding for IMS-based IPTV Home Services*, IEEE Network, 2009
Vidal et al., *Supporting Mobility in an IMS-based P2P IPTV Service: A Proactive Context Transfer Mechanism*, Computer Communications, 2010
Bikfalvi et al., *P2P vs. IP multicast: Comparing Approaches to IPTV Streaming Based on TV Channel Popularity*, Computer Networks, 2011
Submissions in progress

Figure 1.5: The summary of contributions and corresponding publications.

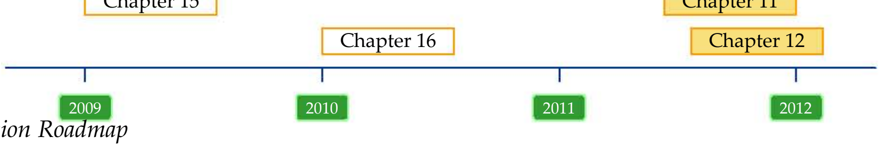
of commercial services is still limited. To this end, by relying on existing data and research work, we take some steps to refine and improve the state of the art (Bikfalvi et al., 2011).

- Fifth, we design two enhancement methods addressing the two main performance issues: signaling delay and peer churn. The fast signaling uses a feedback-based *session coordination algorithm* to estimate the number of inactive uploading sessions that accommodate the viewer demand, while providing an adequate utilization of the network bandwidth. The low churn relies on a *peer coordination algorithm* to manage the peer participation on multiple TV channels and allocate the available bandwidth (Bikfalvi et al., 2009b).

- Finally, we conduct an extensive performance evaluation at every level of our proposal, using modeling and large-scale computer simulations. Our objective is twofold: to identify the performance characteristics of the P2PTV service in a basic setting, as well as to examine a set of realistic scenarios. In one of latter situations, we promote the use of P2PTV as a complement to IP multicast when the peer uploading capacity is limited, and we show that P2PTV constitutes a viable alternative when used for unpopular TV channels (Bikfalvi et al., 2011).

1.4 Dissertation Roadmap

While elaborating this work, one of our fundamental goals was to give a sense of completion by addressing to a certain degree all important aspects from our proposal. At the same time, recognizing the potential complexity of our thesis, we intended it as accessible as possible for the reader, by introducing new concepts in an incremental fashion, and whenever possible explaining



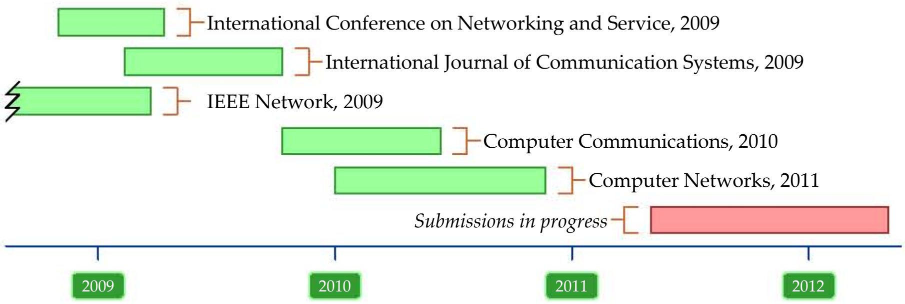


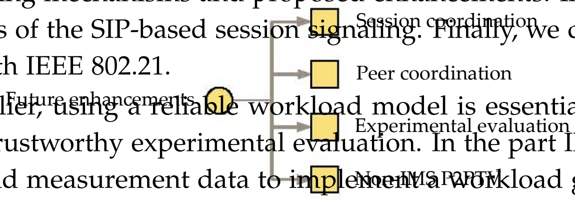
Figure 1.6: The timeline of our work.

the relationships between them. Toward this end, this dissertation is divided into five parts, as shown in the figure 1.7.

The part I, also containing this introduction, sets the scene by providing a brief overview of the P2PTV for the IMS. It underlines the connections to our previous peer-reviewed published work, as to justify the acceptance by the research community of the notions presented here. The next two chapters, 2 and 3, offer a glimpse into the state of the art on IPTV and peer-to-peer video streaming, as well as the standardization effort on next-generation networks and the IP Multimedia Subsystem. We put a special emphasis on the research work related to our own, as well as the current progress on supporting peer-to-peer within the IMS platform.

The part II describes the core concepts of the IMS-based P2PTV. The chapter 4 introduces the service architecture addressing to a manageable all important issues: system integration with the IMS platform, streaming mechanisms and proposed enhancements. In turn, the chapter 5 dives deeper into the details of the SIP-based session signaling. Finally, we devote the chapter 6 to the proactive mobility with IEEE 802.21.

As mentioned earlier, using a reliable workload model is essential for both a performance-driven design and a trustworthy experimental evaluation. In the part III, we rely on the available scientific literature and measurement data to implement a workload generator for a large-scale commercial-grade IPTV system. The chapter 7 extends the existing models, by addressing several omissions and inaccuracies. Subsequently, the chapter 8 describes a tunable workload generation algorithm to simulate real-life operating conditions. In the end, the chapter 9 validates our code against both the initial model and third-party data sets.



We dedicate the part IV to the design of the P2PTV-AS software, which, as the central entity of the service, has a crucial role in the overall performance. The chapter 10 introduces the AS architecture and its main functions, with a special focus on those connected to our proposed enhancements: the session and peer coordination. The chapters 11, 12 and 13 study the session coordination based on a feedback control algorithm, which uses real-time activity data to estimate the near-future demand. In the chapter 14, we present the peer coordination, with the role of assigning the available peer bandwidth to outstanding requests. Unlike the related work, we develop a decentralized mechanism where the peers' own self-organizing behavior assists the resource allocation.

We evaluate the performance of our P2PTV service at two distinct levels. First, the introduction of every new algorithm is followed by a validation phase, which is intended to test the concept, justify the solution and support the progress to the next step. Second, we assess the P2PTV performance from the subscriber perspective. To this end, in the part V, we conduct a set of detailed large-scale simulations to measure the improvements over signaling delay and churn. Whereas the chapter 15 targets a simpler experimental setting, the chapter 16 addresses several practical circumstances such as a hybrid P2PTV-IP multicast approach and heterogenous TV channels.

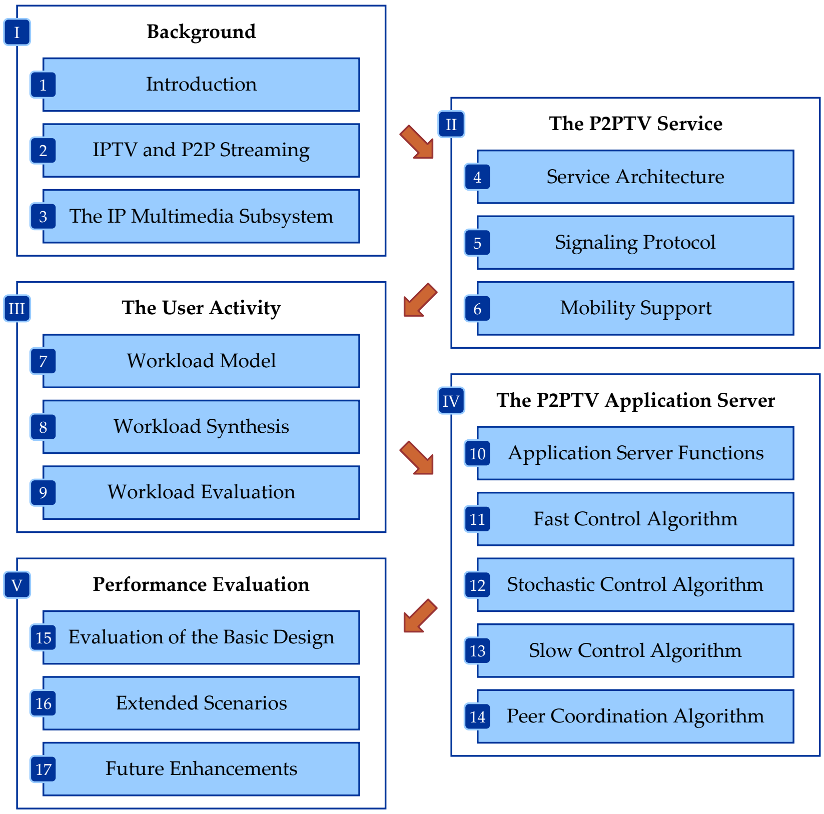


Figure 1.7: The roadmap of this dissertation.

The true research work behind this dissertation is by no means complete. Accepting this, we devote the final chapter to several concluding remarks on the future steps that may refine the P2PTV-AS algorithms even further, may address different experimental approaches, and ultimately may extend the lessons learned beyond next-generation networks and the IP Multimedia Subsystem.

Chapter 2

Peer-to-Peer Streaming and Internet Protocol Television

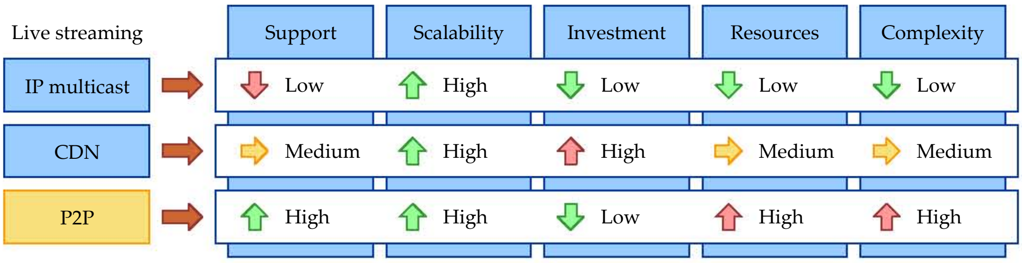
2.1 Overview

During the past decade we have experienced an tremendous growth in the popularity of video content. Current forecasts estimate a threefold increase of the annual Internet consumer traffic by 2016, with the video content having the majority share (Cisco, 2012a, 2012b). This trend brings significant technical and economic challenges on both internet service and content providers. By recognizing the importance the video will play in the near future, peer-to-peer (P2P) has become an increasingly popular solution for the one-to-many delivery of live or on-demand video content, referred to as peer-to-peer streaming.

The P2P streaming ensures scalability with the existing demand, by relying on individual user equipments to contribute their uploading bandwidth. Therefore, it represents a cost-effective alternative to the client-server paradigm or more scalable architectures such as content delivery networks (CDN), requiring little initial investment on the part of the content provider. By considering the prospective opportunities, the P2P streaming for live video has received a lot of attention from both the research community and the industry. We note that while there has been a similar interest for video-on demand (VoD) as well, given the nature of our work, we focus exclusively on the delivery of live content.

Historically, these proposals were justified by the lack of support for IP multicast in the Internet at large. For this reason, the initial efforts attempted to replicate the multicast functionality using the client hosts, or peers, instead of the network routers. The resulting *application-level multicast*[1] or ALM – named in this way because the forwarding of packets is performed at the application as opposed to the network layer of the OSI model – creates and maintains an overlay tree between the participating peers. Once a peer has joined the overlay tree, it will replicate and forward the

[1] Sometimes introduced as application-layer multicast.



Live streaming	Support	Scalability	Investment	Resources	Complexity
IP multicast	⇩ Low	⇧ High	⇩ Low	⇩ Low	⇩ Low
CDN	⇨ Medium	⇧ High	⇧ High	⇨ Medium	⇨ Medium
P2P	⇧ High	⇧ High	⇩ Low	⇧ High	⇧ High

Figure 2.1: A comparison between technical solutions for live video streaming.

downloaded video data to its downstream neighbors, an approach called *push* streaming.

Despite the tremendous opportunity for P2P streaming, justified through high scalability and low deployment cost, its major disadvantage is the difficulty to accommodate the client behavior and the fluctuating network conditions. The former, called churn in P2P terminology, represents the peers leaving the streaming overlay such as when their interest in downloading the live content has ended, or their network connection fails. The latter signifies that the P2P video content competes with the other Internet traffic for the same bandwidth resources, and it is therefore subject to delay and congestion.

Unfortunately, the push streaming solutions have proven too inflexible to handle highly dynamic peer environments, because they would require a fast reorganization of the overlay tree in response to churn or network events. In addressing these issues, the mesh structure emerged as a promising alternative. Commonly referred to in the scientific literature also as *pull* or *data-driven* streaming, with this mechanism the peers do not maintain a specific overlay structure for the video traffic flow. Instead, the peers share their downloaded content, and their neighbors may request specific chunks or segments like in P2P file sharing. The successful delivery in time for playback will be subjected to data and bandwidth availability. Whereas the push methods focus on the management of the overlay tree, the pull approaches target the exchanges of peer information and the scheduling of segment requests.

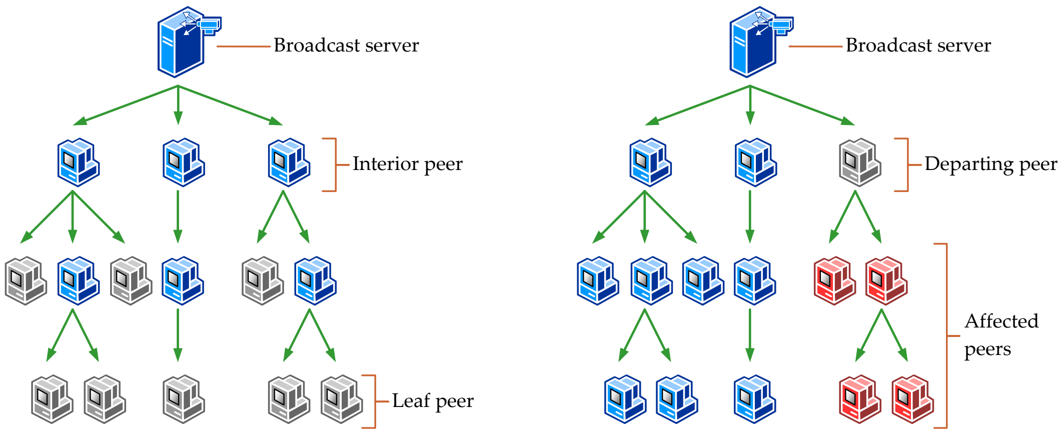
In parallel with the success and growth of the video streaming market in the Internet, an increasing number of telecom operators began offering broadcast television using the Internet Protocol, or IPTV, benefiting from the integration of multiple services over the same packet network. Unlike the Internet video streaming, these commercial deployments target a classic TV viewing experience, with set-top boxes installed at the customers premise and a well-provisioned core network, dimensioned to accommodate the TV traffic (Cha, Rodriguez, Moon, & Crowcroft, 2008).

By carefully considering this distinctions, this chapter offers a brief overview of both video streaming worlds: the Internet and walled-gardens. As already introduced, the former targets P2P proposals, as a necessary background for our work, while for the latter we look at the existing architectural setting, including service features and current business models.

2.2 Peer-to-Peer Streaming

The peer-to-peer has not been the unique answer to the lack of IP multicast. One example is the multicast backbone, or MBONE, that interconnects multicast enabled networks with multicast tunnels (Eriksson, 1994). The tunnels consist of multicast IP packets encapsulated into unicast packets and therefore the multicast traffic can be seamlessly routed. Their disadvantage is that they require manual configuration and stable IP addresses at both ends and, therefore, it is not a feasible solution for home users.

For the delivery of live audio and video, the content distribution networks, or CDNs, consist of



(a) The streaming tree.

(b) The effect of peer churn.

Figure 2.2: Peer-to-peer streaming with application-level multicast.

a hierarchical overlay of distribution servers located closer to the users and with enough bandwidth to support the media streaming session. The multimedia traffic is streamed from the source to some of these servers, which in turn redistribute the traffic both to other servers lower in the hierarchy or to the connected users. This concept uses unicast connections between all participants, and therefore does not require multicast-enabled routers, while relieving the broadcast server from a large number of connections. The traffic load can be balanced between the CDN servers and quality of service can be strictly monitored and enforced. Unfortunately, its disadvantage is the dedicated infrastructure, which represents a significant initial capital expenditure. The protocols used are proprietary in most of the cases with dedicated server and client software.

Finally, the peer-to-peer methods represent a lightweight CDN, where the tasks performed by the servers are transferred to the end-users, replacing the CDN infrastructure with a P2P overlay. The major challenges towards the design and deployment of P2P videos systems are the modest bandwidth capacity of individual peers and the peer churn. Previous research has shown, however, that even in the conditions of the current Internet, P2P streaming is possible on a large scale, though in some cases the uplink capacity is a limiting factor (Sripanidkulchai et al., 2004). To summarize, the figure 2.1 sketches a brief comparison between IP multicast, content delivery networks and peer-to-peer, as possible solutions for live video streaming. We highlight in particular the differences on current network support, scalability, capital and operating costs, and implementation complexity.

2.2.1 Application-Level Multicast

Chu et al. are among the first authors to recognize the need of a viable alternative addressing the deployment deficiencies of IP multicast, such as scalability and management issues, pricing, lack of support for error and congestion control, introducing peer-to-peer as end-system multicast (Chu et al., 2000). Their pioneering work, along with contemporary proposals like Overcast (Jannotti et al., 2000), opened the era of application-level multicast (ALM). The ALM intends to address the needs of any type of multicast traffic by emulating the operation of IP multicast and maintaining a tree overlay between end-hosts.

The figure 2.2(a) illustrates the ALM concept, where the client systems, sometimes referred to as nodes or peers, are responsible for leveraging replicating and forwarding the multicast data, such as audio/video streams. Peers leverage their network connections to receive and send multicast traffic at the same time. Depending on their role in a single ALM tree, we distinguish

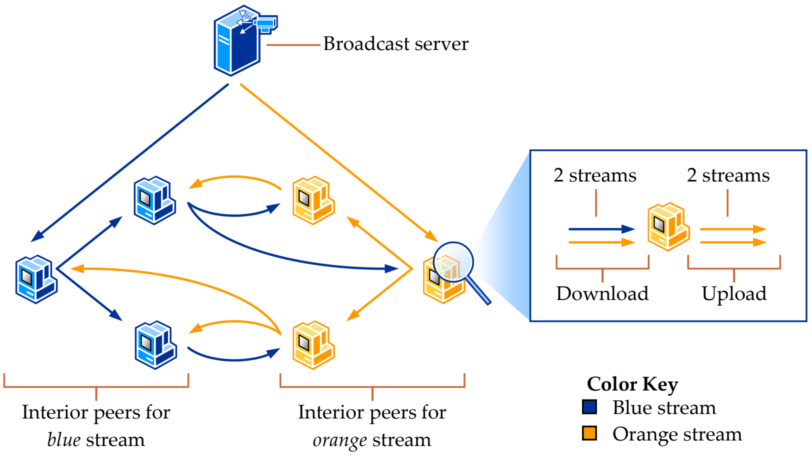


Figure 2.3: Multiple tree push P2P streaming.

between interior and leaf nodes.

The maintenance of ALM tree requires a significant degree of coordination between peers, and to this end, most ALM implementations use a *structured* P2P protocol. This allows all peers to share complete topological information, and create a well-formed, unique and loop free multicast tree. Most structured approaches take the form of a *distributed hash table* (DHT), with properties like routing performance depending on the organization of the hash space and peer routing tables. Among the examples, we mention Chord (Stoica et al., 2001) with a circular hash space, the Content Addressable Network or CAN (Ratnasamy et al., 2001) based on a multi-dimensional coordinate space, Pastry (Rowstron & Druschel, 2001) using a Plaxton-style mesh, and Kademlia (Maymounkov & Mazieres, 2002) employing hierarchical routing based on an *exclusive or* metric.

The major challenge in ALM streaming is maintaining a tree that ensures a reliable service to the participating peers. The departure of a peer, called churn, manifests itself as an interruption of the data flow, affecting the service quality for all downstream peers for the duration of their disconnection, as shown the figure 2.2(b). In targeting these concerns, Castro et al. propose Scribe, built upon Pastry and focusing on the management of the ALM tree (Castro et al., 2002). The traffic flow is described as *push*, meaning that once a peer has become a member of the ALM tree, it does not have to generate further requests for the multicast traffic. Similarly, interior nodes automatically upload the multicast packets to all of their upstream neighbors.

From a performance standpoint, the ALM tree is characterized by a *depth* and a node *fan-out* (i.e. the number of children nodes), influencing the latency and jitter of the multicast traffic. The interior nodes carry all of the forwarding burden, whereas the leaf peers no not contribute their uploading resources. Because many peers only support a limited number of upstream connections, this inflexible situation limits the uploading capabilities of the P2P overlay.

2.2.2 Multiple Trees

By recognizing these drawbacks, the ALM single trees have not been considered serious contenders for video streaming. Instead Castro et al. suggested using several multicast trees and dividing the video stream in a number of stripes or layers (Castro et al., 2003). Their proposal, named SplitStream, argues that the interior nodes from one tree may act as leaf nodes in the other trees, and therefore balancing the peer contribution as shown in the figure 2.3.

SplitStream is more robust in two ways. First, it is less susceptible to churn because the departure of one peer influences only a layer from the full video. Depending on the audio/video codec, the affected peer may continue the playback of the remaining layers, lessening the impact

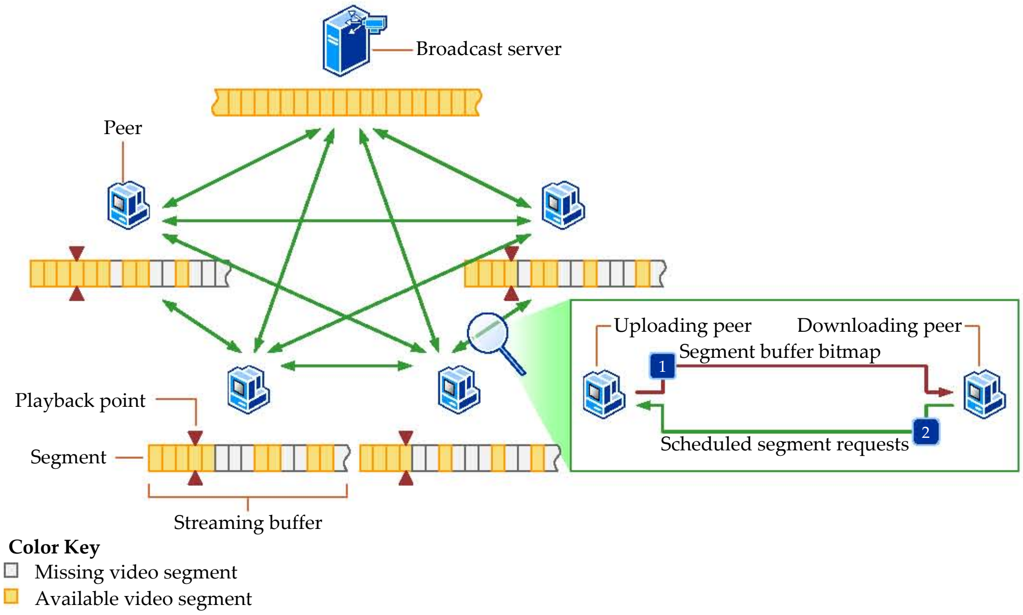


Figure 2.4: Simplified view of mesh, pull-based P2P streaming.

on the viewer quality perception. Second, it increases the bandwidth granularity where interior nodes may serve a layer to a greater number of children as opposed to the full stream.

2.2.3 Data-Driven Streaming

Despite the simplicity of the push streaming with single or multiple ALM trees, many researchers found them inadequate to accommodate the streaming interruptions and missing pieces of video data. In the best effort Internet, there is no guarantee that a peer may support delivery or that a given tree branch will not be affected by congestion. Moreover, the peers have different bandwidth capabilities at different moments of time.

Toward this end, Zhang et al. introduced CoolStreaming/DONet as one of the first streaming proposal to use a *pull* component, where peers may ask their uploading neighbors for specific parts of the stream (X. Zhang et al., 2005). Since then, the power of pull has been adopted by both the research community, with novel algorithms like BiToS (Vlavianos et al., 2006) and PRIME (Magharei & Rejaie, 2009), and commercial implementations such as CoolStreaming[2], PPTV (also known as PPLive)[3], PPS.tv (also known as PPStream)[4], UUSee[5], and many others.

The pull P2P streaming is also called *data-driven* or *mesh*, because the peers exchange information on the available content with many other neighbors. These mechanisms are often implemented using an *unstructured* P2P protocol, where the peers organize themselves according to a random graph. The content discovery may be centralized, using a tracking server, or distributed, based on techniques such as flooding or gossiping.

From a generic perspective, as illustrated in the figure 2.4, the live stream is divided in a number of individual pieces or segments. The peers maintain a streaming buffer, where they store a limited number of video segments. At regular intervals, they exchange with their neighbors the bitmaps of their local buffer to discover the content availability, and subsequently request the missing segments. Both the overlay organization and the segment scheduling play a key role in the performance of data-driven P2P streaming. For instance, the segment scheduling algorithm

[2] http://www.coolstreaming.us/

[3] http://www.pptv.com/

[4] http://www.pps.tv/

[5] http://www.uusee.com/

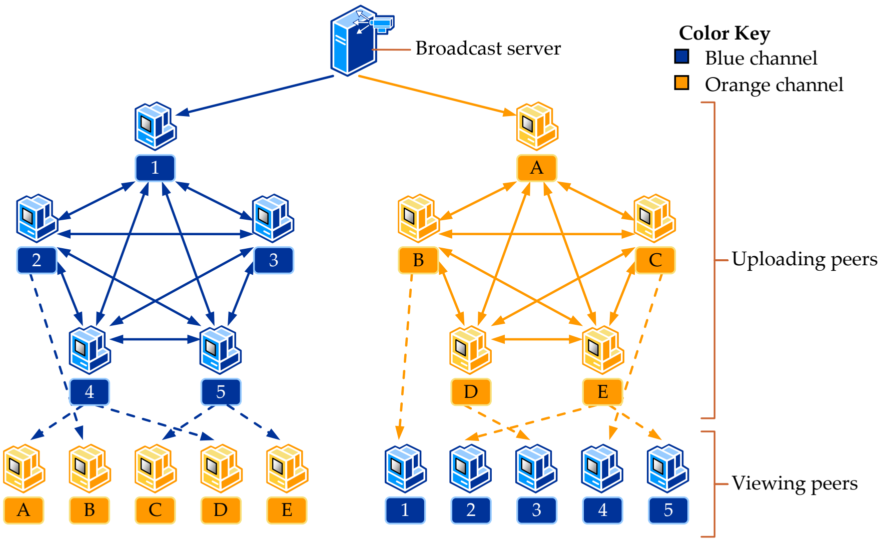


Figure 2.5: View-upload decoupling for two TV channels.

may prioritize certain segments, depending on availability, bandwidth limitation, or playback importance. In this context, Zhao et al. provided an in-depth analysis of the optimal segment selection strategies, based on the trade-off between distribution efficiency and playback urgency (Zhao et al., 2009).

Not all data-driven algorithms implement the previous approach. Examples may included Chunkyspread, which uses multiple trees managed in a swarming-style fashion, as opposed to the structured DHT protocols (Venkataraman et al., 2006). Magharei et al. adopt a hybrid push-pull alternative for their work PRIME, intending to benefit from the flexibility of pull with the low overhead of push, where the peers organize themselves in several trees (Magharei & Rejaie, 2009). During an initial diffusion phase, a video segment is replicated within a tree ensuring a good distribution, then peers in different trees exchange missing segments randomly during a swarming phase.

2.2.4 Multiple TV Channels

With the advent of the first successful P2P streaming implementations, services like PPTV or UUSee began offering multiple TV channels, inching closer towards a classic TV watching experience, referred to as *peer-to-peer television* or P2PTV. Unfortunately, allowing users to browse through a larger TV channel palette has negative consequences on the streaming performance, due to the peer churn amplification generated by the channel changes.

Chuan Wu et al. are among the first authors to recognize the significance of the problem, and they advocate for an enhanced provisioning of server capacity to accommodate the increased peer dynamics (C. Wu et al., 2008). Because using the server does not provide a P2P solution, Di Wu et al. introduce the *view-upload decoupling* or VUD, where the peers download in parallel two streams: one corresponding to the current TV channel requested by the user, and a second for uploading to their neighbors, as shown in the figure 2.5 for two channels. By eliminating the channel-change churn, VUD performs significantly better for heterogenous channels popularity and streaming rate (D. Wu, Liu, & Ross, 2009).

Whereas VUD focuses on the peer participation on two distinct TV channels, more recently Wang et al. improve their work by allowing peers to upload on multiple channels at the same time, given favorable bandwidth conditions (M. Wang et al., 2010). They show this enhanced flexibility yields the best results in terms of the channel bandwidth satisfaction ratio, a measure of the P2P

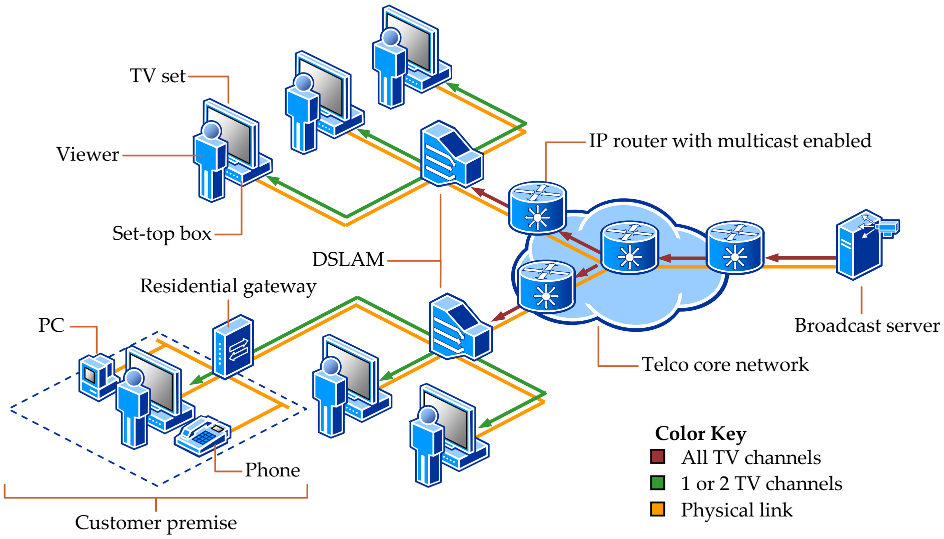


Figure 2.6: A walled-garden IPTV deployment.

overlay to allocate sufficient upload bandwidth with respect to the download demand.

The key tradeoff of these methods for multi-channel P2P streaming is the additional bandwidth used by peers to download the uploading TV channels (or streams), channels that are not consumed by their respective users. Since the problem introduced by P2PTV is the channel-change churn, in parallel to these efforts researchers have analyzed the reduction of the churn rate through an enhanced peer selection. Toward this end, Wang et al. distinguish peer between two tiers according to their overlay stability (F. Wang et al., 2008). They assume the peer lifetime as a reliable indicator of the peer stability, and show, through a number of PlanetLab experiments, that the tiered design gives a significant boost to the streaming quality.

2.3 Walled Gardens

The success of video streaming in the Internet is not unique. Realizing the potential benefits from converging multiple services over the same IP data network, in the recent years an increasing number of commercial providers have began offering television over the Internet Protocol, or IPTV. Unlike the Internet streaming, their service delivers an experience comparable with the classic television, having dedicated set-top boxes installed at the customers' premise that manage transparently the technological differences.

For the best customer satisfaction, the majority of these IPTV deployments employ on a dedicated network infrastructure, specially dimensioned to accommodate the expected video traffic. Referred to as walled-gardens, these networks are controlled by the telco and they are mostly restricted to outside traffic. They address congestion by over-provisioning, and the video streaming uses IP multicast. In one of the first measurement study of these commercial IPTV environments, Cha et al. provide a glimpse into their architectural details (Cha, Rodriguez, Crowcroft, et al., 2008). As illustrated in the figure 2.6, the service uses static IP multicast to stream all TV channels from the broadcast server, or IPTV head-end, to the edge of the network, usually a DSLAM or a point-of-presence. The last-mile connectivity, between the telco and the household subscriber, supports one or two TV channels simultaneously, and the channel change limits the interaction between the household set-top box and the border router, guaranteeing a fast response.

However, in a subsequent publication the same authors recognize that many of these deployments have been limited to only a few hundred TV channels, and future growth of the available TV would be unsustainable under the same design philosophy (Cha, Rodriguez, Moon, & Crowcroft,

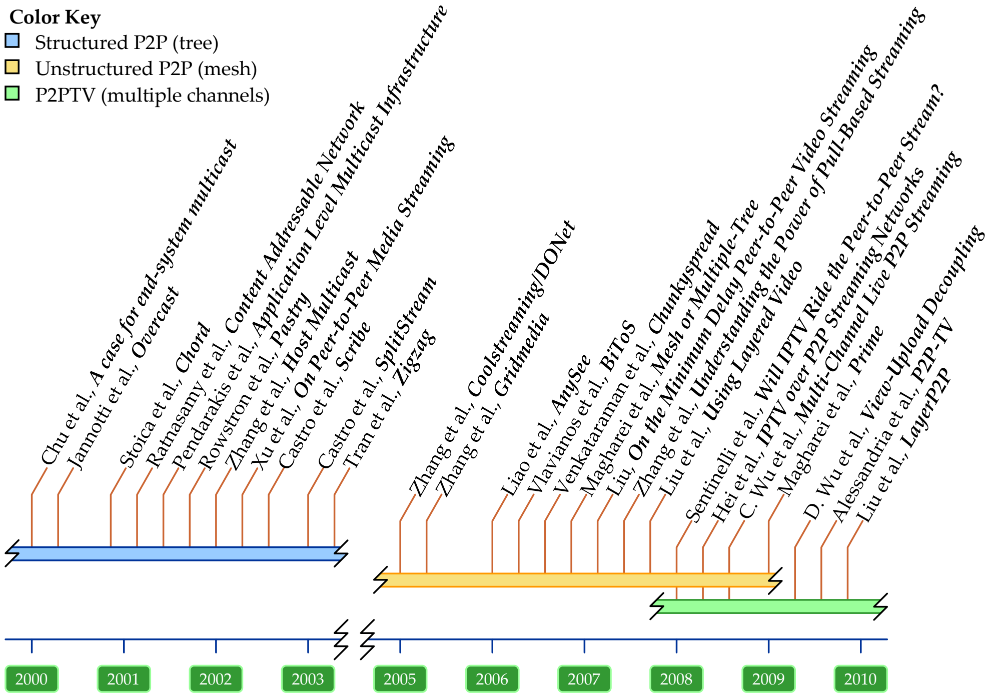


Figure 2.7: Key milestones of the P2P streaming related work.

2008). Toward this end, they argue for P2P streaming as a complementary solution, underscoring that the telco-managed environment offers a unique opportunity for enhanced performance, which is not available in Internet-based applications and where the IPTV set-top box provides an equal level of transparency and user experience.

2.4 Conclusions

During the past decade, the growth of video content popularity in the Internet has triggered a significant effort from both the industry and the research community in developing novel cost-effective approaches for the distribution of live content. Despite the Internet Protocol built-in support for multicast, the deployment is still limited over twenty years later after its introduction. In the search for a viable alternative, the attention shifted from the network routers to the end-systems with the role of replicating and forwarding the multicast data, including video streaming.

As illustrated by the key milestones from the figure 2.7, in the early days P2P streaming emulated IP multicast by using one or more multicast trees. Despite their elegance, the tree approaches exhibit poor performance in real-life due to the difficulty of managing the tree structure in highly dynamic peer environments. Instead, the newer data-driven or mesh streaming focuses on the successful delivery of the video data, as the ultimate indicator of the system performance. Its success had been proven by several implementations, leading to a new class of commercial P2PTV online services that have evolved toward a traditional TV watching experience. In this context, the support for multiple TV channels emerged as an important new issue, with current solutions promoting multiple channel participation to mitigate the channel-change churn.

In parallel, service providers have also adopted Internet Protocol streaming replacing their legacy infrastructure. These networks, specially dimensioned to accommodate the TV traffic, use IP multicast to stream TV channels to the household subscribers. However, some researchers argue that in the future P2P streaming may prove a complementary approach, when the large number of available TV channels may be inefficiently managed by IP multicast alone.

Chapter 3

Next-Generation Networks and the IP Multimedia Subsystem

3.1 Overview

During the last decade, there has been a clear trend in the policy of telecom operators to converge their legacy networks into a unified platform. This move comes for reasons of cost efficiency, as well as an attempt to facilitate their entrance in the market of multimedia services, until now strongly dominated by the Internet industry. In this context, the next-generation network, or NGN, began as an umbrella term for the evolution towards an integrated IP broadband network, capable of supporting a wide variety of multimedia services. Its goal is to allow transport providers, namely telcos or ISPs, to integrate easily their services with those of third parties.

The work of the 3rd Generation Partnership Project (3GPP) represented one of the first efforts towards standardizing a specification for NGN. The core of their proposal is an IP Multimedia Subsystem (IMS) designed to manage virtually any type of multimedia service, while providing the main features of the NGN goals (3GPP, 2012c). These include easy service implementation, support for quality of service (QoS), seamless mobility, a common infrastructure for authentication, policing and charging. The NGN decouples the service provision from the transport network through a set of standardized interfaces, with the goal of providing easy integration to third party services, and where the core service control platform is the IMS.

Given the historical origins of the 3GPP work on the third generation mobile communications, the IMS standards put a special emphasis on the interaction with UMTS networks. Recognizing this, the Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN) working group of the European Telecommunications Standards Institute (ETSI) have been extending the 3GPP specifications towards a more universal NGN, capable of supporting a wider range of access technologies, as well as interacting with legacy networks (TISPAN, 2008). In addition, the TISPAN efforts the diversity of multimedia services. In this respect, recognizing the increasing popularity of live and on-demand video content in the Internet, they standardize an

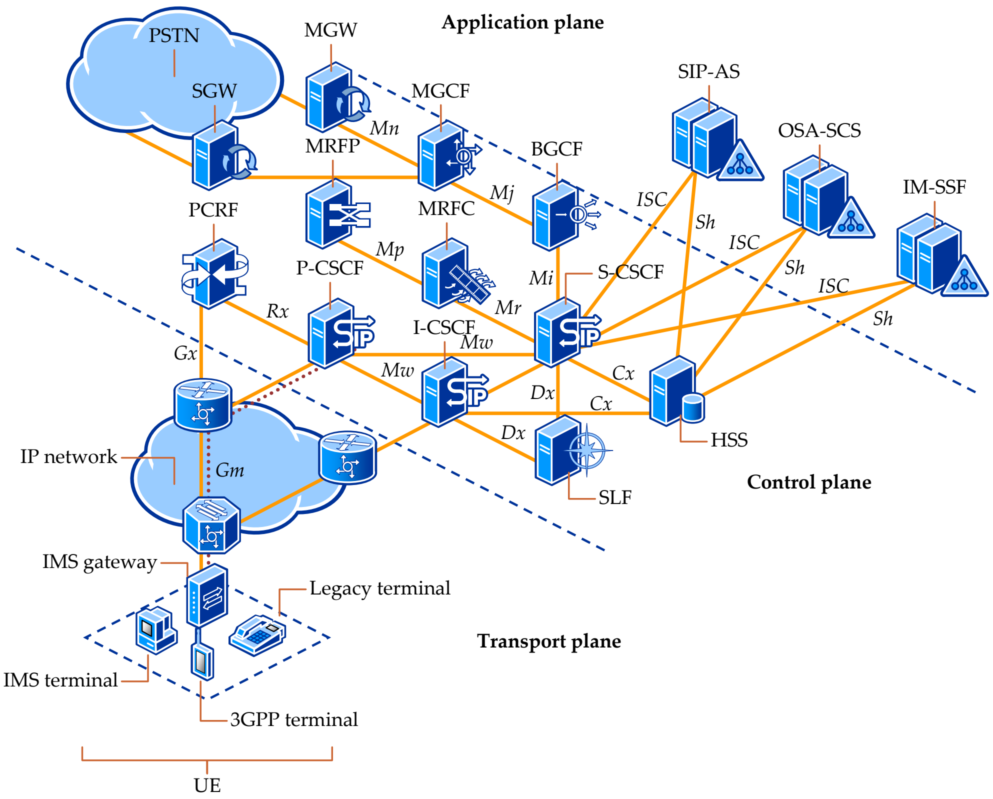


Figure 3.1: The architecture of the IP Multimedia Subsystem as defined by 3GPP.

IPTV architecture for the NGN (TISPAN, 2011a).

It is worth mentioning that 3GPP and TISPAN are not alone in their efforts of defining a common IP infrastructure for multimedia services. Whereas 3GPP and ETSI are European bodies, in parallel, 3GPP2 in the United States has had a similar agenda but with a focus on the evolution of the third generation CDMA2000 standards. However, given the worldwide popularity of the UMTS and subsequent mobile technologies, like LTE, in the present work, we target exclusively the 3GPP and ETSI specifications.

3.2 The IP Multimedia Subsystem

The IP Multimedia Core Network Subsystem (IMS) is a control architecture based on the Internet Protocol that enables the provision of IP multimedia services to the end-users. The IMS consists of a number core network functional entities, interconnected across standardized interfaces or *reference points*, and it includes support for many next-generation network features such as establishing IP multimedia sessions, quality of service, roaming, security and charging. The IMS implements interworking requirements with existing legacy networks and devices.

The figure 3.1 illustrates the simplified architecture of the IMS. From a higher perspective, the architecture follows a layered approach, where we identify three planes: a *transport* plane or sometimes referred to as *user*; a *control* plane, and; an *application* plane. This organization facilitates the separation between the transport networks, the session management functionality at the control plane, and ultimately the implementation of multimedia applications. In other words, in a fashion similar to the role of the Internet Protocol, the IMS accommodates different access technologies to distinct services with a single control framework.

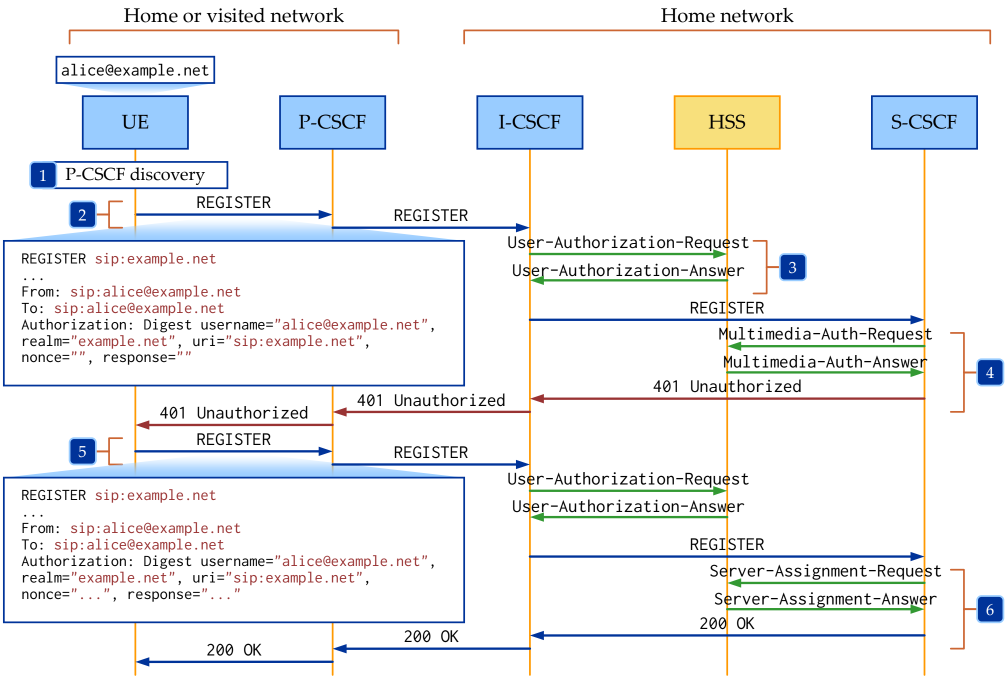
Ultimately, the goal of the IMS is to support the transmission of multimedia data between service participants, such as the voice traffic between two users engaged in a phone call, or the

Name	Acronym	Description
Proxy CSCF	P-CSCF	It is the entry and exit point into the IMS control plane for every SIP signaling message originating or terminating at the UE. It is located in the home or visited network of the UE, depending on the type of access technology.
Serving CSCF	S-CSCF	It performs user registration and session control by checking the incoming SIP requests against the user service profile, and routing them towards their destination: another user equipment or an application server (AS). It is located in the home network.
Interrogating CSCF	I-CSCF	It is the entry point of an administrative domain, and discoverable via DNS. It is usually located in the home network.
Home subscriber server	HSS	It is a database containing the IMS subscriber service profiles. It is accessed by the S-CSCF during the initial subscriber registration, and it may be accessed by the application server to handle custom service requests.
Subscriber location function	SLF	It is a database that identifies the HSS corresponding to a given user identity, when there are multiple HSS in the home network.
Policy control and charging rules function	PCRF	It provides policy control decision an flow-based charging functionality (3GPP, 2012f).
Breakout gateway control function	BGCF	It provides routing for SIP messages addressed to circuit-switched networks, by forwarding the messages to another BGCF or a selected MGCF.
Media gateway control function	MGCF	It is a gateway between the IMS and other circuit-switched networks, translating the SIP signaling to the protocol used by the circuit-switched network.
Media gateway	MGW	It is an interface between the IMS and the circuit-switched network in the media plane.
Signaling gateway	SGW	It is an interface between the IMS and the circuit-switched network in the signaling plane.
Multimedia resource function controller	MRFC	It is the signaling component of the multimedia resource function (MRF), responsible for the handling of media content such playing announcements (3GPP, 2012d).
Multimedia resource function processor	MRFP	It is the user plane component of the MRF.

Table 3.1: The IMS functional entities from the control plane.

stream between a video content provider and its subscribers. The designers envisioned IMS as the evolution towards platform that unifies various types of communications, and therefore all multimedia traffic uses IP packet-switching. Unlike in the Internet, before communicating party may begin exchanging data they must establish a multimedia session, which, like the signaling of a phone call, ensures the user has the permission of using the service, allocates the necessary QoS resources, implements the charging policy, etcetera.

The Session Initiation Protocol (SIP) (Rosenberg et al., 2002) standardized by the IETF is the session control protocol in IMS, and its selection was due to its flexibility of implementing new services (Camarillo & Garcia-Martin, 2011). In addition to SIP, the IMS relies on many other protocols, most of them specified by the IETF. One example is Diameter (Loughney et al., 2003) used for authentication, authorization and accounting or AAA. The IMS subscribers interact with the platform via their devices, generically referred to as user equipments (UE). These may be mobile terminals like a smartphone or a tablet, or fixed such as a computer or a residential gateway,



```
REGISTER sip:example.net
...
From: sip:alice@example.net
To: sip:alice@example.net
Authorization: Digest username="alice@example.net",
realm="example.net", uri="sip:example.net",
nonce="", response=""
```

```
REGISTER sip:example.net
...
From: sip:alice@example.net
To: sip:alice@example.net
Authorization: Digest username="alice@example.net",
realm="example.net", uri="sip:example.net",
nonce="...", response="..."
```

Figure 3.2: The signaling procedure of the initial user registration.

acting as a proxy for many other non-IMS equipments.

3.2.1 Functional Entities

The core session control functionality of the IMS is implemented by a number of *call session control functions* (CSCF), which process the SIP signaling of the multimedia sessions. There are three types of CSCFs, depending on their specific role over the session control. A database called the *home subscriber server* (HSS) stores the user information, known as profile, which includes its identity and the list of subscribed services. Other functions implement the policing and charging requirements, which validate the reservation of network QoS resources during the session establishment. The table 3.1 briefly summarizes the main entities from the core IMS. The protocol used between pair of functional entities is specific to each reference points, summarized in the table 3.2.

Protocol	Reference points					
SIP	*Gm*	*ISC*	*Mi*	*Mj*	*Mr*	*Mw*
Diameter	*Cx*	*Dx*	*Gx*	*Rx*	*Sh*	
H.248	*Mn*	*Mp*				

Table 3.2: The core IMS reference points and the corresponding protocol.

Outside of the core IMS, the application servers (AS) host and execute the multimedia services. An AS is a SIP server interacting with the multimedia SIP dialog on behalf of the service provider. Depending on the specifics and functionality of each individual service, the AS may act as the terminating party in the multimedia session, originating party, or proxy. Both the IMS standards and the SIP specifications leave a high degree of flexibility at the latitude of the service developers as to the specific role the AS plays in the session control. The SIP-AS is the native application server for IMS services. For interworking purposes, the Open Service Access - service capability server (OSA-SCS) and the IP multimedia service switching function (IM-SSF) enable access to CAMEL (3GPP, 2011b) and OSA (3GPP, 2010) services, respectively.

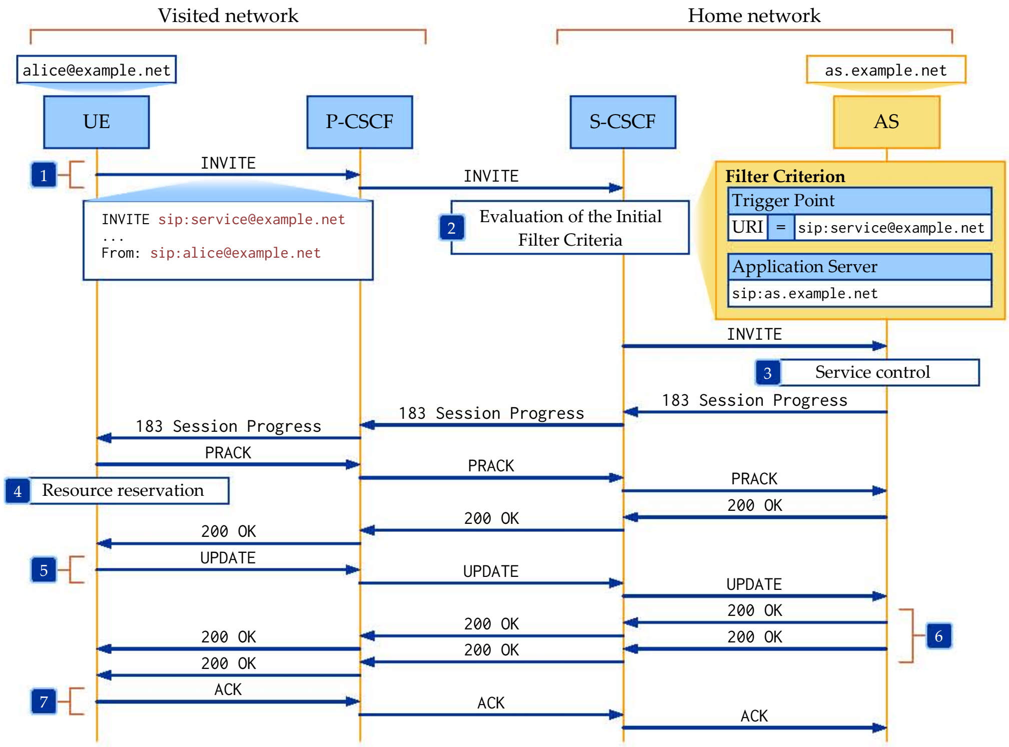


Figure 3.3: The signaling procedure of a session establishment with a SIP application server.

3.2.2 Service Control

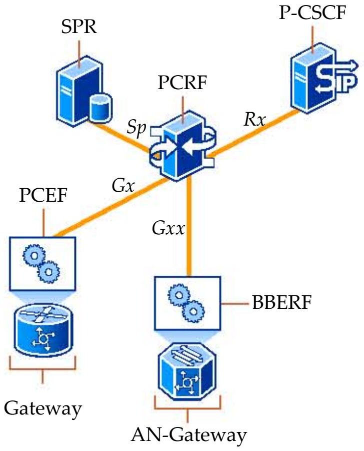
All IMS subscribers have one or more service profiles stored at the HSS, indicating the multimedia services they may access from their user equipment. Each profile corresponds to a pair of private and public user identity. The private identity taking the form of a Network Access Identifier or NAI (Aboba & Beadles, 1999), such as `alice@example.net`, is reserved exclusively for user authentication. The public identity, a SIP Uniform Resource Identifier (URI) like `sip:alice@example.net`[1], is used for the SIP routing and it is included in all SIP messages during the signaling of a particular session.

The user authentication, and therefore the validation of its identity, takes place during the IMS *registration*. This procedure is executed when the UE connects to the network for the first time. The figure 3.2 sketches, from a simple perspective, the signaling steps involved. Without entering into greater details, the UE initialization begins during the step 1 with the discovery of a P-CSCF, usually located in its current (home or visited) network.

The UE initiates the registration procedure by sending a `REGISTER` message to the P-CSCF with the request URI set to the domain name of the home network. The request includes a `From` and a `To` headers containing the public user identity, and an `Authentication` header with a structure that depends on the current authentication scheme. In our example, assuming the digest authentication from RFC 2617 (Hallam-Baker et al., 1999), the header includes a `username` field containing the private user identity, and empty `nonce` and `response` fields, and a `realm` and a `uri` fields set to the domain name and SIP URI of the home network, respectively (step 2).

The P-CSCF forwards the registration request to the I-CSCF, which uses the HSS to authorize whether the user may register in the current network, to discover and forward the message to the S-CSCF assigned to the user (step 3). Upon receiving the request, given the empty `nonce` and `response` field from the `Authentication` header, the S-CSCF replies with a `401 Unauthorized`

[1] In addition to a SIP URI, the public user identity may also be a TEL URI (3GPP, 2012e).



Legend

P-CSCF	Proxy Call Session Control Function
PCRF	Policy and Charging Rules Function
SPR	Subscription Profile Repository
PCEF	Policy and Charging Enforcement Function
BBERF	Bearer Binding and Event Reporting Function

Figure 3.4: The policy and charging architecture.

response that includes a nonce challenge for the digest authentication (step 4). The UE uses the challenge to re-initiate the registration a second time, where the request `Authentication` field includes a response (step 5). Assuming the response is correct, the S-CSCF completes the registration with a `200 OK` response.

While the UE is registered to the IMS, it may initiate multimedia sessions according to the services set in its current service profile. These sessions are established with another UE, or with the application server of a particular service provider. Given that the signaling exchange in both situations is similar, and considering the setting of our work, the figure 3.3 illustrates the procedure for the second case, where the multimedia session terminates at the AS.

In this example, the UE establishes a multimedia session to a service represented by a *public service identity* `sip:service@example.net`. The UE initiates the dialog by sending an `INVITE` SIP message with the request UE set to aforementioned service identity (step 1). Although not shown in the figure, the request body may include a Session Description Protocol (SDP) payload containing the session information, such as the media type, IP address and transport port, QoS requirements, etcetera (Handley et al., 2006). The request is forwarded to the S-CSCF via the P-CSCF.

Upon receiving the session request, the S-CSCF processes the message against a set of filters stored in the user's profile. These filters, formally named *initial filter criteria* (or iFC), represent a mapping between one or more conditions or *triggers* and a forwarding destination. Considering the iFC from the figure, the S-CSCF forwards the request to the AS (step 2). The application server performs the service control, which in this instance is completing the establishment of the session (step 3). By assuming that the initial `INVITE` carried a number of QoS preconditions that must be met before returning a final response, the AS replies with a `183 Session Progress` provisional response informing the UE of the dialog progress.

The UE returns a provisional acknowledge and initiates the reservation of QoS resources required to fulfill the session preconditions (step 4). After the resource reservation completed successfully, the UE initiates an update of the dialog containing the current QoS (step 5). Finally the AS completes the session establishment by confirming both the `UPDATE` and the initial `INVITE` with a `200 OK` response (step 6), acknowledged by the UE (step 7). At this instance, the UE and the service provider may begin exchanging multimedia data, according to the session description agreed at the signaling time. We note that while the dialog is in progress, any party may modify the session with an `UPDATE` request.

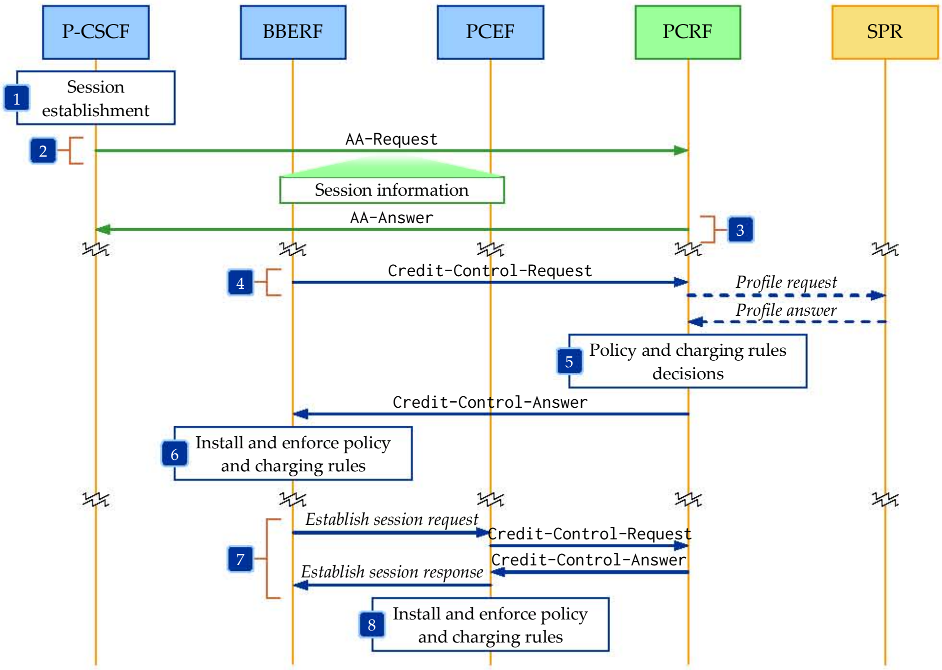


Figure 3.5: Policy and charging during the session establishment.

3.2.3 Quality of Service

One of the key features of the IMS is the built-in support for quality of service. As shown in the previous example, the QoS resources reservation is an essential component of the session establishment. At a lower level, QoS provisioning requires the installation of traffic filters at the appropriate network routers and gateways, such that they allow the session data and process it according to the QoS parameters.

From a functional perspective, the QoS is managed by the policy control and charging (PCC) functions. The figure 3.4 shows these components in greater detail, including their interaction with the routers or gateways from the transport plane. The primary functional entity in the PCC architecture is the PCRF. As shown in the figure 3.5, during the session establishment (step 1), the PCRF receives from the P-CSCF a Diameter `AA-Request`, over the *Rx* reference point, containing the session information derived from the SDP payload (step 2). The PCRF uses the policy from the *subscription profile repository* – a database containing subscription-based PCC rules – to determine the QoS filters required for the current session.

The UE initiates the resource reservation, usually by contacting its access network (AN) gateway, although the actual procedure is specific to the access technology. The AN gateway implements a *bearer binding and event reporting function* (BBERF), and all gateways implement a *policy and charging enforcement function* (PCEF). Upon the trigger received from the UE, the BBERF contacts the PCRF, via the *Gxx* reference point, to determine, install end enforce the policy rules for the session in progress (steps 4–6). The procedure is repeated on behalf of other gateways from the transport network (steps 7–8).

This brief overview of the signaling procedures underscore that, despite the obvious benefits offered by QoS provisioning in IMS, the installation of the QoS rules is an expensive process requiring a high degree of interaction between various functional entities. This is true in both at the session control level, and in the policy and charging architecture.

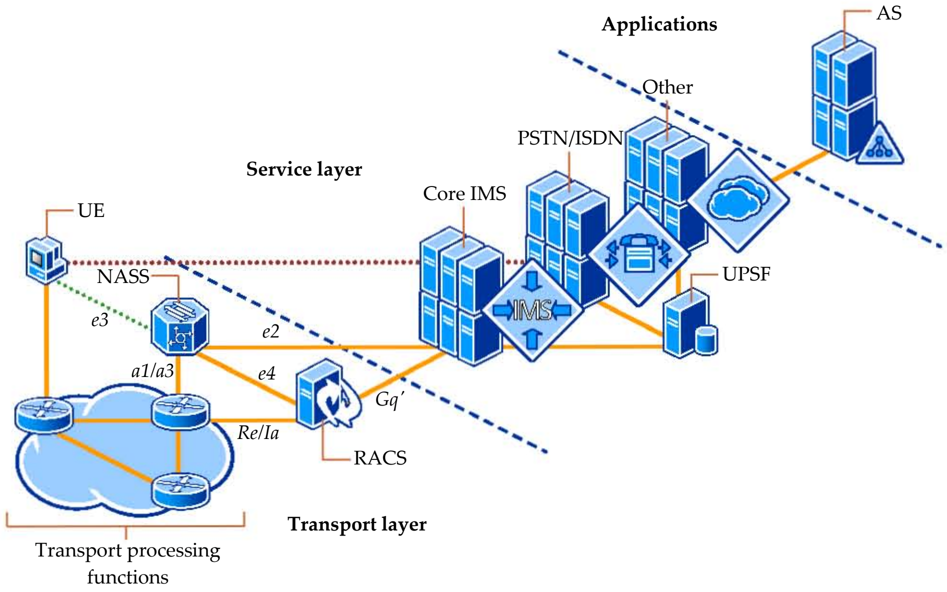


Figure 3.6: The simplified architecture of a TISPAN next-generation-network.

3.3 IPTV in the IP Multimedia Subsystem

Alongside the standardization effort from 3GPP, which focused mainly on mobile networks, the Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN) working group of the European Telecommunications Standards Institute (ETSI) have been releasing a parallel set of specifications for a next-generation network, based on the IP Multimedia Subsystem core network. While many of the standard aspects mirror the work of 3GPP, there are several essential distinctions.

The figure 3.6 illustrates the equivalent NGN architecture in TISPAN (3GPP, 2007). By analyzing the differences, the focus on convergence of the TISPAN NGN becomes clear. In addition to the IMS, the architecture supports multiple legacy platform, such as the public switched telephone network (PSTN). The same approach is following in the transport plane, designed from start to adopt multiple access technologies. Instead of the PCRF, the resource and admission control subsystem, or RACS, is responsible for resource reservation, policy and admission control to provide support for quality of service (QoS) (TISPAN, 2011b). The RACS hides the transport network details from the applications, while allowing operators to enforce a network-wide policy on how resources are allocated and used. The user profile server function (UPSF) is the equivalent of the home subscriber server from the 3GPP specifications.

The flexibility and universal nature of the TISPAN next-generation network, based on the IMS core network, has triggered many studies on IMS-based services, including IPTV. Alongside proposals like (Xiao et al., 2007; Más et al., 2008; Mikoczy et al., 2007), the 3GPP and TISPAN have been working in completing a specification for IMS-based IPTV (TISPAN, 2011a). In the figure 3.7, we show the architecture of the IPTV service for NGN defined in TISPAN. The specification introduces a number of new functional entities divided in two categories: service functions and media functions (MF). The service functions enable the service discovery (SDF), service selection (SSF) and service control (SCF), where the SDF and the SCF are SIP application servers. The media function is divided between the media control (MCF) and the media delivery (MDF) functions.

The specification divides the IPTV services into two main categories: broadcast service (BC), that is live TV streaming, and content-on-demand (CoD). The BC service uses IP multicast to deliver the content, where each TV channel is assigned to a multicast group. In the media plane,

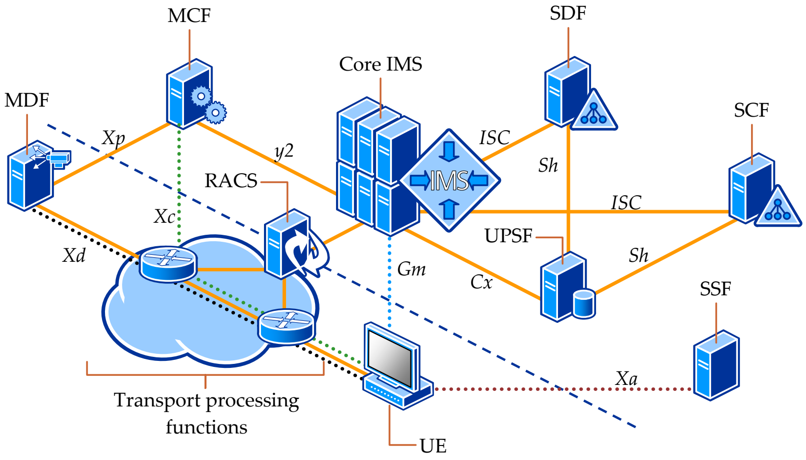


Figure 3.7: The simplified architecture of an IPTV service in a TISPAN next-generation network.

connecting or switching to a channel involves joining or leaving a multicast group using the Internet Group Management Protocol (IGMP).

Recently, TISPAN also began researching peer-to-peer technologies for content delivery for IPTV services. Although their work is still at the draft status, they focus on various peer-to-peer delivery architectures including merging peer-to-peer and content delivery network technologies into hierarchical structures. This new approach, clearly different from the classic IP multicast solution, recognizes that peer-to-peer can be a future solution for resource intensive applications such as IPTV or video-on-demand (VoD), at least in use cases where the media is delivered asynchronously or interactively.

3.4 Conclusions

The next-generation network (NGN) is an integrated broadband network using the Internet Protocol, designed to act as a flexible platform for any type of multimedia service. The most widely used implementation is based on an IP core network called the IP Multimedia Subsystem, developed by 3GPP. The architecture of an IMS-based NGN has a layered structure, divided between a transport plane, a control plane and an application plane. The first is responsible for physical connectivity and bearer access, the control plane manages the service session, and the application plane implements the services access by the subscribers.

In parallel to the 3GPP specifications, which focused primarily on mobile access, the TISPAN group of ETSI standardized a complementary NGN architecture, which is extended to many other bearer technologies. As part of their efforts, TISPAN has recognized the crucial role of IPTV services, and released an IPTV architecture for the next-generation networks. They introduce a number of functional entities, designed for both live and on-demand video content.

In this chapter, we introduced a brief overview over the architecture of an IMS-based NGN, as defined by both 3GPP and TISPAN. We examined the main functional entities, and we insisted on key signaling aspects that we considered relevant from the perspective of our work. Among these, are the requirement of SIP-based signaling when establishing a multimedia session between communicating parties. Equally important, the QoS provisioning comes with the cost of a high degree of interaction between the functional entities in the control and transport plane.

Part

II

A P2PTV Service for the IP Multimedia Subsystem

Chapter 4

Service Architecture

4.1 Overview

The peer-to-peer television service for the IP Multimedia Subsystem, abbreviated in this work as P2PTV or P2P-IMS-TV, provides a common platform for IPTV streaming using the peer-to-peer technology. Its role is to replace classic video streaming techniques such as IP multicast or content delivery networks (CDN) in situations where they are not administratively or economically viable. The idea behind P2PTV is to exploit the content upload potential of the customer premise NGN-enabled devices, such as residential gateways (RGW) or set-top boxes (STB). However, given the centralized nature of the IP multimedia subsystem, where all multimedia sessions are coordinated through a set of core functions, developing a distributed and P2P-oriented service represents a significant challenge. In particular, our solution must accommodate all standardized specifications of the IMS in terms of session establishment and service control.

Consistent with the typical business model of the IMS-based next-generation-networks (NGN), the service supports the peer-to-peer streaming mechanism independent of the IPTV content providers, allowing multiple providers to take advantage of a larger common set of streaming resources. A reverse separation is also possible, where an IPTV provider uses the P2P platform only for a subset of its available TV channels. While throughout this work we hint upon the possible business model for the P2PTV service, we acknowledge that other deployment options are available, including the one where each individual IPTV provider uses its own P2P platform. For this reason, our main focus is the technical aspect of the peer-to-peer TV channel streaming in the IMS, leaving open the integration aspects.

In an IPTV service for the IMS, the end-user devices, commonly called in the IMS terminology user equipments (UE), receive the TV channels from one or more broadcast servers. As in any IMS-based NGN, the access to the IPTV service is done at two different planes: the *signaling* or *control plane* that performs the IMS-specific functions, and the *media* or *transport plane*, which handles that actual delivery of the IPTV content. The figure 4.1 illustrates the integration of an IPTV service with the IMS, and emphasizes the difference between the signaling and media

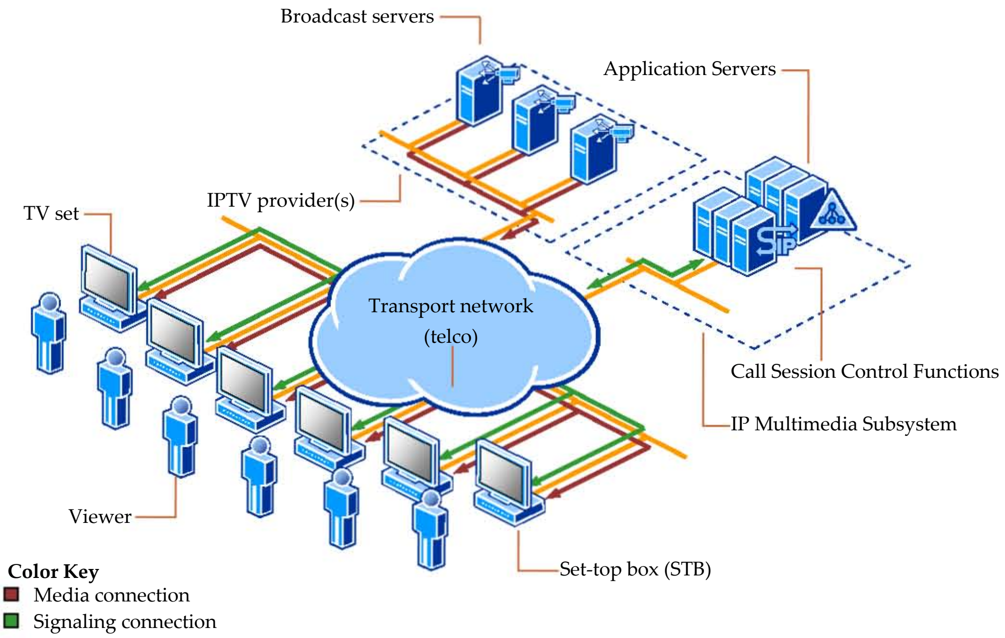


Figure 4.1: System overview: IPTV as an IMS service.

connections used over the same physical infrastructure.

In an IMS-based NGN, the TV channels are streamed in the context of a multimedia session established between the subscriber user equipment (UE), such as the set-top box, and the broadcast server (BS) of the content provider. The underlying network protocol is IP, and the existing standards allow both unicast and multicast connections. From the perspective of our work, the main role of the IMS signaling is the reservation and commitment of access *network resources* that are essential to a high quality service delivered to the user. In this context, by network resources we understand mainly network bandwidth and class of service.

As we mentioned previously, from a functional perspective the architecture of an IMS NGN is divided into a *transport plane*, a *control plane*, and an *application plane*. To this end, every multimedia sessions between two entities in the transport plane is first signaled in the control plane. This allows the validation of the session or access to the service based on the user profile, applies the network policy to reserve and commit resources, perform charging and many other functions.

An IPTV service usually provides a specific SIP application server (AS) that processes the session requests from the user equipments on behalf of the service provider. While the IPTV broadcast server (BS) and the corresponding application server (AS) can be the same physical equipment (similar to an IPTV UE, which includes both transport and control functions), in this work we refer to them from their functional perspective where the BS is located in the transport plane, while the AS in the application/control plane[1]. In addition to its function as a SIP user agent, the AS may provide enhanced IPTV service features such as the available TV channels, program guide, video recording functions, etcetera.

When the AS implements the TISPAN IPTV specifications, it can be identified with the Service Discovery Function (SDF) and the Service Control Functions (SCF) described by the standard. The figure 4.2 emphasizes the mapping between the AS and BS, the corresponding TISPAN functional entities, and the interfaces between them. Since our proposal neither require nor focuses on the implementation of the all features from the TISPAN standards, for the remainder of this work we shall refer simply to the AS and BS.

[1] Some authors prefer the separation of an IMS NGN between transport and control planes, with the AS included in the latter.

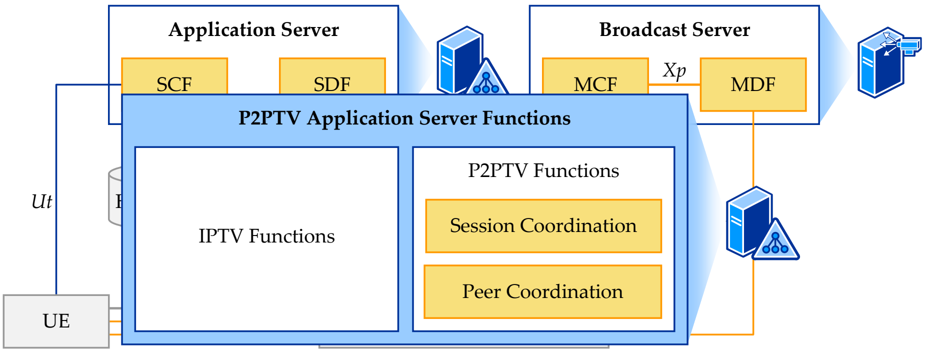


Figure 4.2: Mapping of the IPTV application server (AS) and broadcast server (BS) to the TISPAN functional entities.

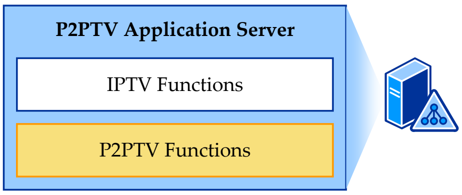


Figure 4.3: The P2PTV-AS combines the traditional IPTV-AS and functions directed at the P2P streaming.

4.2 System Architecture

The central element of our P2PTV service is a SIP application server, which we call a P2PTV application server or P2PTV-AS. Whereas in the traditional IMS IPTV service the IPTV-AS connect the end-user devices to one or more configured broadcast servers, in P2PTV the user equipments upload the TV channel streams to the devices of other viewers. Therefore, our P2PTV-AS performs a number of additional functions, especially related to the UE participation. As we emphasize in the figure 4.3, it is this last component of the P2PTV-AS that constitutes the focus of the present work.

As the reader can well imagine, the architecture of the P2PTV service is more complex than the operation of the P2PTV-AS alone, although the AS can be regarded as the most important. To begin, in the figure 4.4 we put the system architecture in perspective, by showing the integration between the various physical components[2].

4.2.1 The User Equipment

The user equipment (UE) is the IMS-capable device with which viewers access the TV channels. The UE can be a fixed equipment, such as a set-top box (STB) or a residential gateway (RGW), or a mobile device, such a tablet or a hand-held. In this particular instance, we show two types of UEs: a set of STBs using xDSL connectivity and several UMTS wireless devices. These examples are illustrative only, and it is possible to use many other devices or access technologies. However, we intend to emphasize that the P2PTV service may use both wired and wireless UEs that are supported by the IMS. Because UEs may both download and upload the video streams of their current TV channels, we shall refer to them as *peers*.

Unfortunately, at present, wireless bandwidth is still an expensive resource, especially when we

[2] The system architecture uses the 3GPP version of the IMS standards, which provides support for UMTS connectivity, but the main elements of our architecture would remain the same in the TISPAN version.

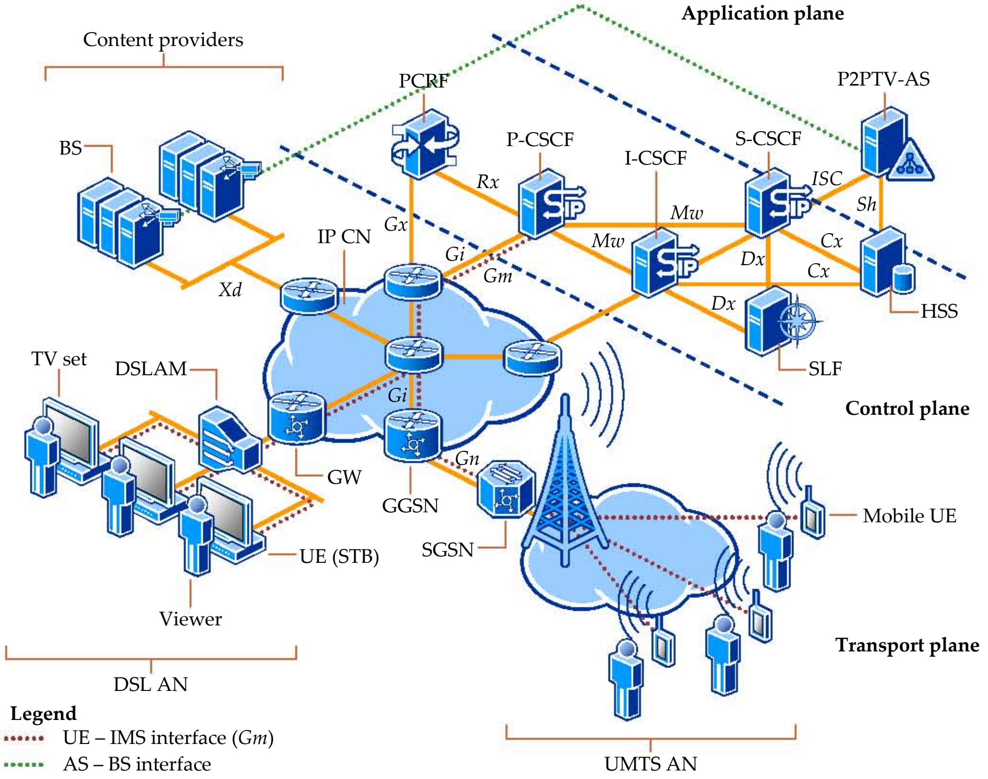


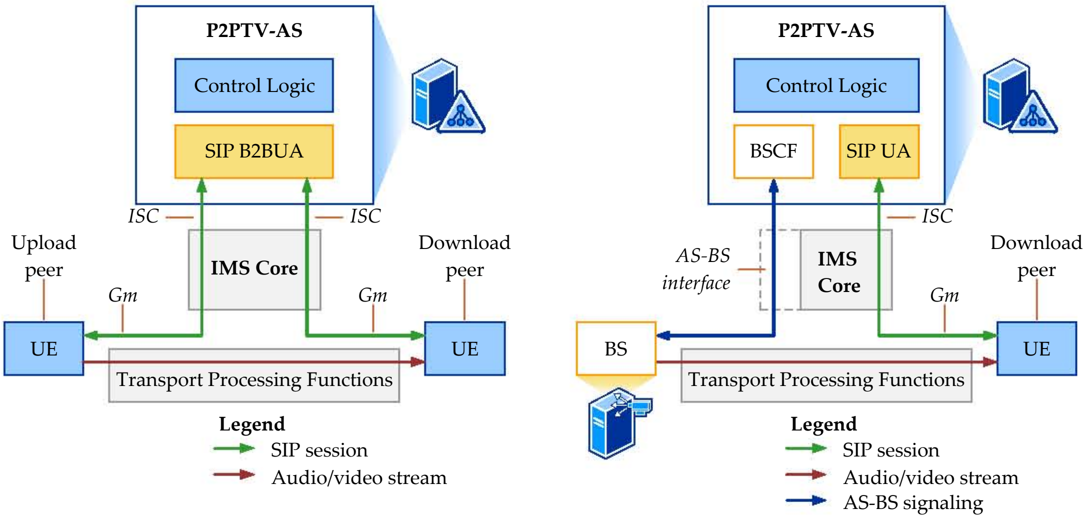
Figure 4.4: The system architecture of the P2PTV service.

look from the perspective of bandwidth-hungry applications such as video and audio streaming. As a witness, despite the present ubiquity of wireless gadgets, virtually all existing commercial IPTV deployments use wired/fixed access networks. A P2PTV service that uses UEs as content uploaders and downloaders simultaneously puts a significant strain on connections that already have limited bandwidth. In addition, unlike server-based IPTV, P2P requires UEs to maintain connectivity to many other peers. Since mobile UEs may seamlessly roam between networks changing their IP connectivity configuration, designing the P2PTV service for wireless access requires a specific approach. For these reasons, our main work, including many design decisions and experimental setup for performance evaluations, focuses on fixed UE devices such as STBs, and we devote the chapter 6 to analyze and propose a solution for mobile environments.

At the control plane, the UE uses the *Gm* reference point to communicate to the assigned P-CSCF. Every UE registers to the IMS and connects to the P2PTV service by using an appropriate subscription in the user profile. For dedicated IPTV devices, such as set-top boxes, these steps are executed automatically at start-up, without user intervention. In general, the UE interface presents to the viewer options for the access of the TV channels, while hiding the IMS and/or the P2P operation. For instance, a STB UE may permit the user to access the TV channels using a remote control, similar to non-IMS IPTV technologies.

4.2.2 The P2PTV Application Server

The P2PTV-AS is the SIP application server of the P2PTV service provider. Its main objectives are to process incoming TV requests from the UEs, and to manage the peer-to-peer streaming, where selected UEs both download and upload the TV streams. In order to enable the P2P connections, while making the nature of the streaming sessions transparent to the UEs, the P2PTV-AS is a SIP back-to-back user agent (B2BUA), capable of mapping the signaling sessions between two UE



(a) Between two peer UEs.

(b) Between the BS and an UE.

Figure 4.5: Overview of the multimedia session management by the P2PTV-AS.

peers that communicate to each other. The P2PTV-AS communicates to the S-CSCF in the IMS core via the *ISC* reference point, and it may access the user profile in the HSS/UPSF via the *Sh* interface.

The figures 4.5(a) and 4.5(b) illustrate the role of the P2PTV-AS in the P2P session management. When the system uses P2P streaming between an uploading and a downloading peers, the multimedia session is established between UE and the P2PTV-AS, via the IMS core network. This approach has the advantage that all UEs in the system communicate only to the Public Service Identity (PSI) of the P2PTV service, such as `sip:p2ptv@example.net`, and do not to learn the identities of their corresponding peers.

The B2BUA communicates internally to the *control logic* of the AS to setup the mapping between sessions. As an intuitive example, when the AS receives a new session request for a selected TV channel stream (such a SIP session would be originating at the UE and terminating at the AS), it uses the control logic to determine the identity of an uploading peer that could service the request. The B2BUA initiates a new session to the uploading UE, and logically it links the two sessions together such as sessions changes at one side are reflected at the other. The chapter 5 describes the further detail the operation of the AS B2BUA for the most important signaling procedures.

When the P2PTV-AS cannot use P2P streaming, for instance when the first users connect to a TV channel, it uses the provide's broadcast server. As we show in the figure 4.5(b), while the server-peer sessions resemble closely to the peer-peer case, there are a few subtle differences. First, SIP is not necessary between the P2PTV-AS and the BS, as both entities are connected directly to the transport core network and we assume that the content provider already provisioned the necessary network resources. Second, the signaling may or may not be routed through the IMS core network, depending on the service configuration and the technical agreement between the P2PTV and the content provider (3GPP, 2012c). In this work, we leave open the implementation of the *AS-BS interface*, and assuming the AS implements a generic broadcast server communication function (BSCF), we shall simply refer to it in this manner. When both the AS and BS conform to the TISPAN specifications, the interface is *ISC* between the AS and the IMS, and *y2* between *IMS* and the *BS* (TISPAN, 2011a). Alternatively, if the P2PTV and the content provider are the same entity, the P2PTV-AS and the BS may be the same physical equipment, and the AS-BS interface can be implemented in software alone.

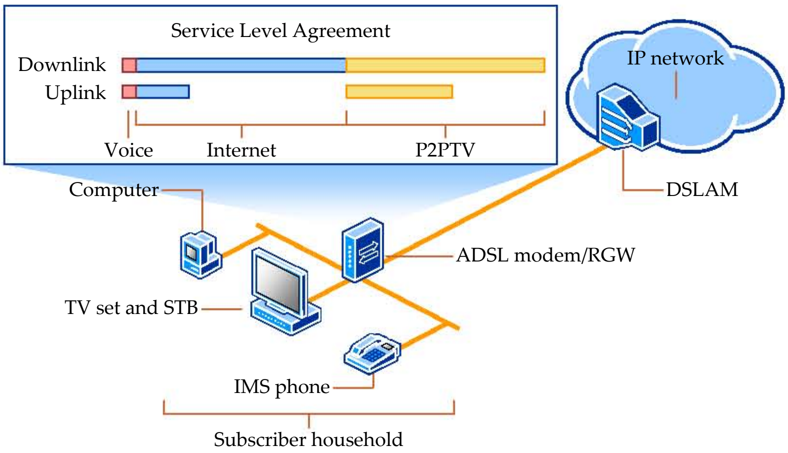


Figure 4.6: The allocation of access network bandwidth to the P2PTV service.

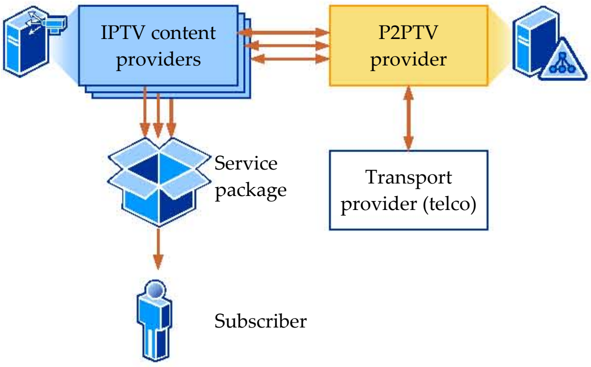


Figure 4.7: The business model of the P2PTV as an NGN service.

4.2.3 Business Model

As in many commercial IPTV deployments, the P2PTV service does not interfere with other communication services subscribed by the user over the same access connection. In the typical triple-play package, the user may contract telephony, Internet access and television, where they may all be IMS services or not. The figure 4.6 illustrates a hypothetical service-level agreement (SLA) for this scenario, where the access network bandwidth of an ADSL subscription is shared between voice, Internet and the P2PTV services. The SLA is contracted between the transport provider (telco) and the individual service providers.

The rationale behind the P2PTV service is to benefit from the unused network capacity, consisting of deployed bandwidth that is not contracted by the user. To this end, one could imagine an ADSL2+ access network that provides roughly between 16 Mbps and 24 Mbps, in the downlink direction, but household subscribers contract Internet packages of 6 Mbps or 10 Mbps. P2PTV is a flexible option that exploits unused network resources, in that it does not require a constant or minimum contribution from the existing user equipments: a UE peer may participate with either more or less bandwidth, depending on its current service contract.

When all services are 3GPP IMS-based, the allocation of all QoS resources is done at the network's Policy and Charging Rules Function (PCRF) and is enforced by the Policy and Charging Enforcement Function (PCEF) implemented at the network's routers. Similarly, in a TISPAN IMS, the QoS resources are managed by the equivalent Resource and Admission Control Subsystem, or RACS.

In the NGN business model, the customers benefit from the wider range of services. In general

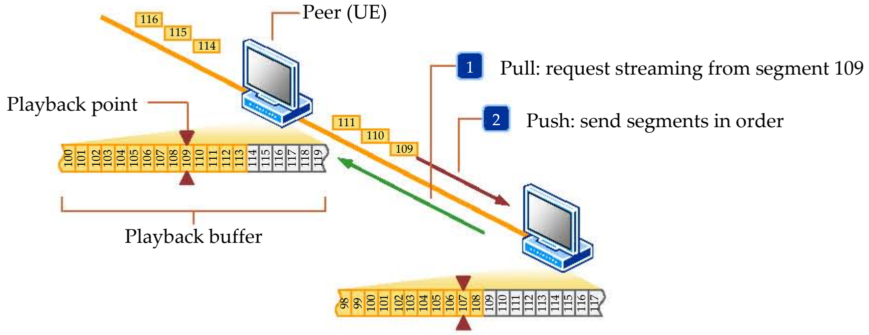


Figure 4.8: Pull-push streaming for the P2PTV service.

they can maintain a contractual agreement with a *service packager* entity, such as the telco, which in turn manages the agreements with the other service providers. The service providers establish relationships with the transport provider as needed, although this is not necessary when the packager is the transport provider. In this context, the P2PTV service is a middle layer between one or more TV content providers and the network operator, as we illustrate in the figure 4.7 (Bikfalvi et al., 2009a). As we mentioned before, this approach has the benefit of reducing complexity at the UE, while allowing multiple content providers access to a greater pool of resources. The P2PTV service contracts from the telco the unused capacity of its access network, while the IPTV providers pay the P2PTV operator for the P2P access.

4.3 Streaming Architecture

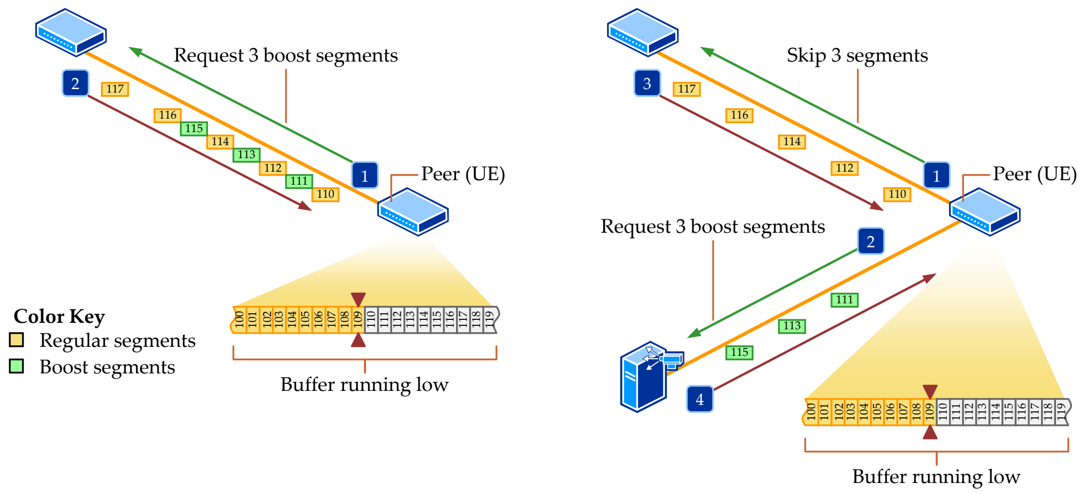
In the media plane, each TV channel is divided into an integer number audio/video *streams*. This division is necessary for two reasons. First, many access technologies, such as ADSL, do not provide the UE peers with sufficient upload bandwidth to support the uploading of a complete TV channel to many other peers; in many cases even the upload of a single channel is not possible. Second, by decreasing the quantum of bandwidth reservation, the available bandwidth are utilized more efficiently, increasing the overall performance of the P2P streaming (Castro et al., 2003).

4.3.1 Push vs. Pull Streaming

The existing scientific literature on P2P video has analyzed in great depth the differences between *push*-based or *pull*-based streaming. The push approach has been associated often with application layer multicast (ALM), where peers forward the received stream continuously to their downstream counterparts, generating a tree-like structure. In pull, the peers discover and download individual pieces of the video independently from many other neighbors, based on their current needs, forming a mesh.

The differences and similarities between the two P2P streaming methods still represent a hot debate among researchers (Magharei et al., 2007; M. Zhang et al., 2007), and we described them briefly in the chapter 2. The pull or mesh techniques have proved very successful in commercial deployments, due to their content-driven and self-organizing nature, which handles better peer churn and fluctuating network conditions. However, their disadvantage is the greater startup delay necessary for the initial buffering. On the other hand, tree approaches have remained of interest mainly in the P2P research field (Hei et al., 2008).

Moving from the Internet to an IMS-based NGN requires us to reconsider which streaming technique is appropriate for out P2PTV service. Mesh streaming works well when a peer communicates constantly with many others, exchanging buffer maps and requesting video segments.



(a) On the same multimedia session.

(b) On a new multimedia session.

Figure 4.9: Streaming boost using the pull-push approach.

Because in an IMS NGN all peers must establish a SIP session for any type of communication out of the IMS signaling band, it is more difficult for peers to organize and exchange data in a swarm-like fashion. In addition the large startup delay introduces a serious performance penalty, amplified by the TV channel changes. Furthermore, because QoS resources are reserved and guaranteed in an NGN, the rationale for selecting the mesh is less justified, as the media exchanges does not have to accommodate varying network conditions, but only the peer churn. For these reasons, the P2PTV service must adopt a tree streaming mechanism.

Although tree streaming is less flexible that the mesh on the exchange of the audio/video content, we can draw elements from the latter to improve the overall performance. One of these elements, which we mentioned before, is increasing the video data granularity and robustness to peer churn, by providing multiple streams per TV channel. Second, the exchange of video data may use a hybrid, *pull-push*, approach where peers do not act as simple forwarding nodes. Because each peer must buffer part of the video stream for its own playback, downstream peers may request a stream upload starting at a given point on the video timeline. This performance enhancement, illustrated in the figure 4.8, mitigates playback gaps or interruptions by ensuring that peers have a continuous stream of data in their buffer.

A pull-push method is useful to synchronize multiple streams of the same TV channel when beginning the playback. The segmentation of the video streams may be done at various sizes, such as frames, group-of-pictures, etc., depending on the video encoding, desired buffer memory and acceptable buffering delay. Peers keep track of the current segment they upload to each of their downstream neighbors. Unlike the Internet implementations, applying the pull-push to IMS P2PTV does not require video segment scheduling at both sending and receiving peers, as all peers must have the bandwidth guaranteed by the network. However, under unusual circumstances, such as when running low of a certain stream, the UE peers may require a *stream boost*, either by asking their sender to send the segments at a faster rate, if the resource reservation supports it, or establishing a new session only to download a number of boost segments.

The figures 4.9(a) and 4.9(b) illustrate the principle of a pull-push streaming boost. During push streaming, when the stream buffer is running low based on the current playback pointer, the UE may request additional segments to fill the buffer. In the first case, if the resource reservation provisions a little additional bandwidth, the boost segments are transmitted over the same multimedia session. Otherwise, in the second case, the UE established a new temporary session,

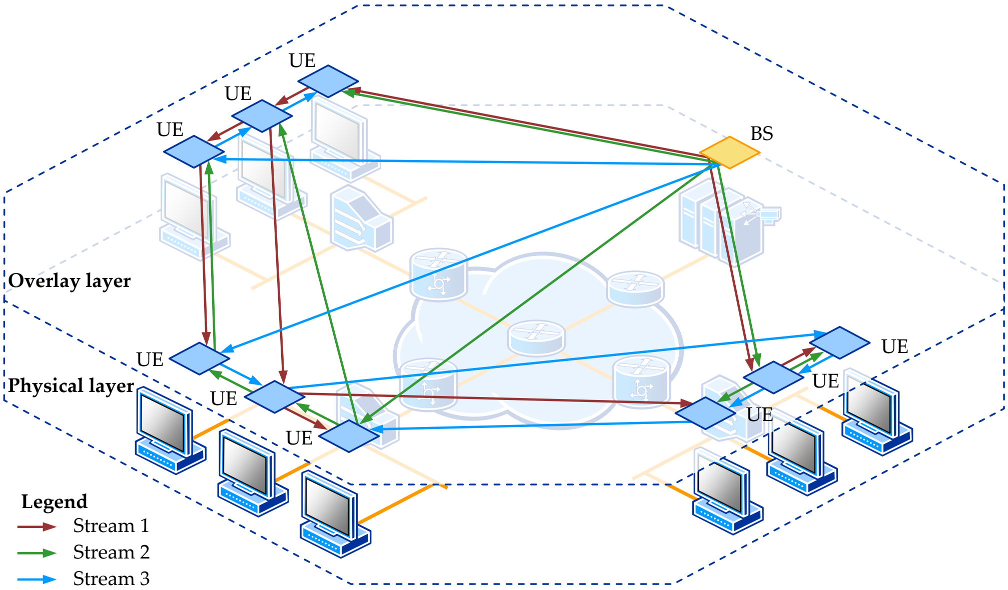


Figure 4.10: Streaming overlay over the physical connectivity.

with the same or a different uploading entity, to receive the boost segments. In principle, because the playback rate matches the streaming rate, streaming boost is only used to accommodate an immediate action, such as a channel change or peer churn.

4.3.2 Channel Streaming

When the UE peers tune into a TV channel, they download all the corresponding audio/video streams. At the same time, the peers may upload one or more streams to other *neighboring* peers, using the pull-push mechanism we described previously. The set of logical streaming sessions between peers generates a *streaming overlay* over the physical network infrastructure, consisting of a number of streaming trees, one for each channel stream. In this context, neighboring peers are always the peers that share a logical link in the overlay, and not necessarily geographical neighbors.

The figure 4.10 illustrates the overlay concept for a TV channel divided in three streams. In this setting, we show only the peers connected to the selected channel, emphasizing each stream and the associated tree rooted at the broadcast server. Dividing the TV channel in several streams enables a greater peer participation, where all UE set-top boxes may upload one or more streams, as opposed to having only a fraction of peers uploading one or more times the whole TV channel.

In practice, there are several approaches to dividing the video into multiple streams. In a naïve way and depending on the video encoding, may be divided at various temporal levels, such as individual fraction of frames, frames, or groups-of-pictures[3]. The drawback of the naïve method is that the TV channel can only be decoded if all streams are received successfully. The annex G of the H.264/AVC standard provides a more robust alternative to the encoding and transmission of video in multiple streams (ITU-T, 2011). Named as *scalable video coding*, or SVC, it allows the partial reconstruction of a video using fewer streams than the complete set at a relative quality that is greater than the fraction of the decoded streams.

SVC provides three different types of scalability: *temporal*, *spatial* and *quality*, where the last can be regarded as a special type of spatial scalability (Wiegand et al., 2003). Temporal scalability, illustrated in the figure 4.11, uses different frame types in a group-of-pictures to encode the stream

[3] A group of successive frames that share a coding relationship.

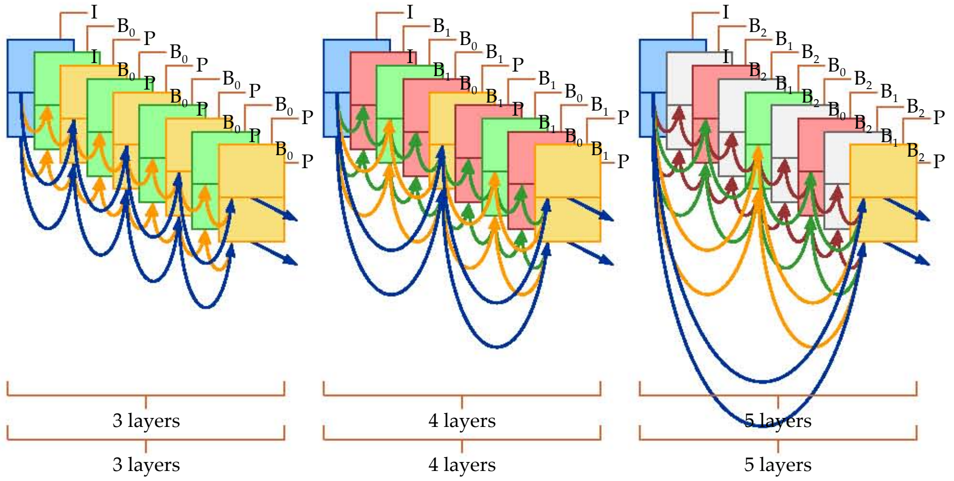


Figure 4.11: SVC temporal scalability with hierarchical coding in a group-of-pictures.

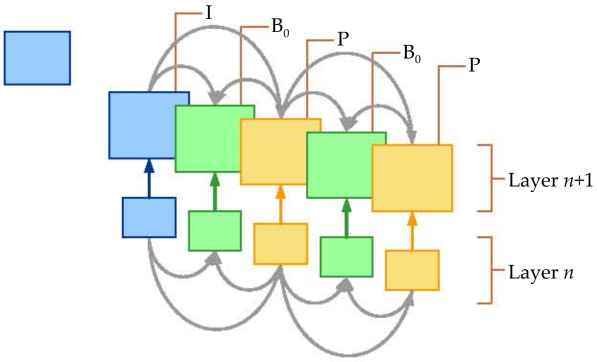


Figure 4.12: SVC spatial scalability with hierarchical coding in a group-of-pictures.

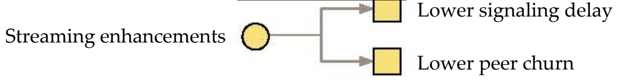
layers. If the encoding within the GOP is hierarchical, as illustrated in the figure, the decoding of certain layers, such as *B*, is conditioned by receiving the *I* and *P* frames first. Many temporal SVC structures are possible, depending on the coding relationship between the frames and the size of the GOP. The low-delay encoding structures, that feature only forward prediction, are in particular useful for P2PTV, where channel changes corresponding to a large fraction of user activity.

Spatial scalability, as shown in the figure 4.12, uses different layers to encode the TV channel at a progressive higher resolution. In principle, the video is divided into a *base* layer and one or more *enhancement* layers. Each enhancement layer contains information that permits the picture upsampling from the lower layer. In addition to the upsampling, spatial scalability permits the enhancement layers to contain additional inter-layer prediction data, such as prediction of macro-blocks, of partitioning, of motion information, etc. Finally, quality scalability uses additional enhancement layers to increase the signal-to-noise-ratio (SNR) of the base layer.

4.4 Streaming Enhancements

While the peer-to-peer video streaming has been of interest to researchers for more than a decade, examining the problem in a more complex setting, where multiple channels are offered at the same time, has only received attention in the past years. In addition, the deployment of such a solution in a next-generation-network poses additional design challenges, because the advantage of guaranteed quality of service comes at the cost of a complex signaling procedure making the establishment of connections between peers an expensive operation.

We address these performance issues in two ways.



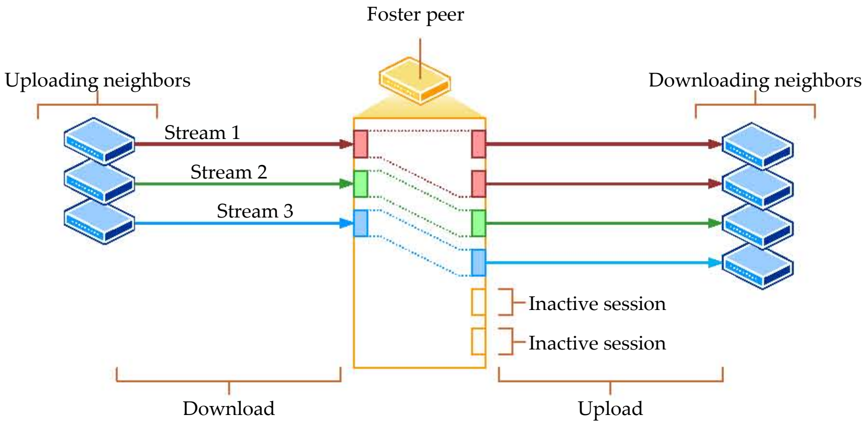


Figure 4.13: The concept of foster peer and inactive sessions.

- The first is done in the control plane, by minimizing the signaling delay during the session setup.
- The second targets the reduction of peer churn, by extending their stream participation beyond their normal channel session.

4.4.1 Fast Signaling

The quality of service advantage of IMS-based next-generation networks comes at the cost of resource reservation and commitment at the time of session establishment. Given the number of entities involved in the setup of a transport bearer with guaranteed QoS, the delay during session setup can be significant. Fast signaling is the reduction of the SIP signaling steps between two peers during the establishment of a multimedia session. Although in the chapter 5, we describe the signaling protocol in greater depth, at the moment our main objective is the elimination of the service provisioning and resource reservation at the time a downloading peer connects to an uploading peer.

To this end, we introduce *foster peers* that have established but *inactive* uploading sessions to the network, and which can begin uploading only with a session update. By shifting the upload session signaling, and especially the expensive resource reservation procedure, before downloading peers connect, we ensure that the P2PTV service is ready to accommodate new users. The name *fosters* suggests the adoption of orphaned peers when their current uploading neighbors leave causing churn, or when they change to a new TV channel.

The foster peers and inactive sessions are provisioned only on the uploading side of the peer-to-peer overlay. On the downloading side they are not necessary because the UE peers reuse the downloading sessions when changing from one channel to another. Therefore, they are generally established once, during the initialization of the UE, and are maintained during the entire online session. The inactive sessions are stream-based, and every peer may have one or more on the streams that is currently downloading. They are multimedia sessions established with the P2PTV application server, which have reserved QoS resources but no associated media stream.

The figure 4.13 demonstrates the concept with the example of a foster peer that downloads the streams corresponding to the TV channel from the previous overlay example. The peer uploads some of the streams to other neighbors and maintains several inactive sessions for future upload connections. To obtain a system with a greater flexibility, we assume that the streams for all TV channels have similar bit-rate requirements. As shown in the previous section, encoding techniques such as SVC may accommodate several layering solutions, making possible the division of the video in a number of streams that meet our requirement. Under these constraints, the

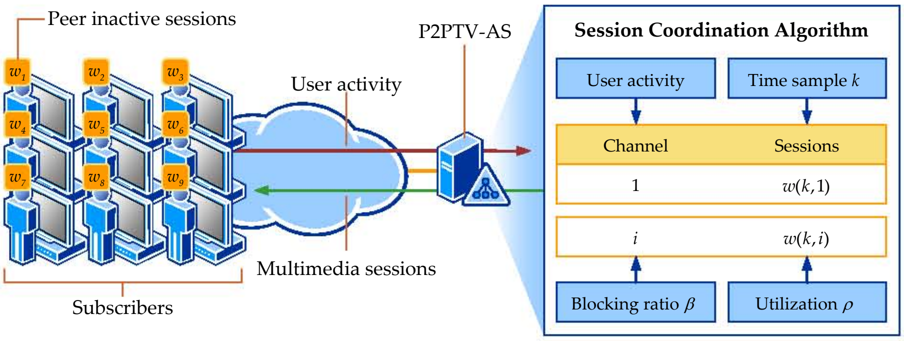


Figure 4.14: The principle of the P2PTV-AS session coordination algorithm.

inactive sessions may be used to upload any stream received by the peer. This aspect is illustrated in the figure, where the inactive slots do not share the color of the other streams.

The management of the inactive sessions is done at the P2PTV-AS. The number of inactive sessions is a per-channel property and varies with time. The P2PTV-AS is responsible to compute, for every TV channel and at regular time intervals, the *necessary* number of inactive session. The AS initiates the establishment of the inactive session, when the existing number is less than the computed one, or the release of some sessions, when the existing number is greater. By necessary number of sessions, we understand the number that is sufficiently large to accommodate the user demand for the given channel, but low enough such that it does not waste the network resources.

Computing the number of inactive session is a complex operation. In principle, as we illustrate in the figure 4.14, the AS measures the user activity on each TV channel, based on the incoming requests, and uses a *session coordination algorithm*, or SCA, to estimate the near future demand for the same channel. The estimated number of sessions, per channel, is denoted by w, and the number of inactive sessions represent the difference between the computed value and the actual viewers count.

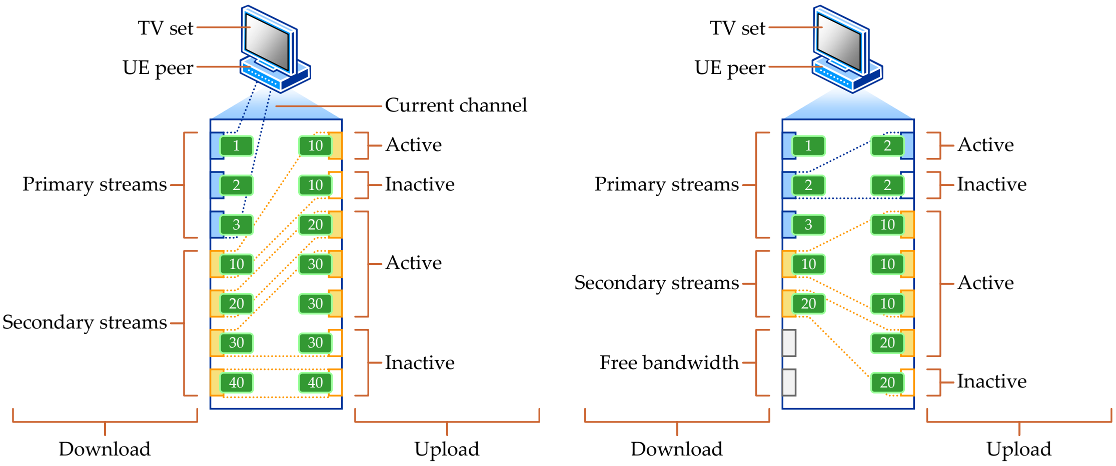
The SCA uses two performance indicators to determine the number of sessions that offer the best compromise between servicing new requests and the conservation of network resources. The desired blocking ratio, denoted by β and selected by the operator, represents the maximum allowed fraction of new channel requests that may not be accommodated with an inactive session. This fraction of sessions is similar to the blocking probability concept from telephony and queueing theory, except that in our situations these sessions are not blocked entirely from receiving service, but instead experience a longer establishment delay. The session utilization, ρ, is the fraction between the active and the total number of sessions. If we denote the total number of upload sessions by w, active sessions by w_a, and inactive sessions by w_i, then we have:

$$\rho = \frac{w_a}{w_a + w_i} = \frac{w_a}{w}. \tag{4.1}$$

The operation of the session coordination algorithm requires an extensive understanding of the user activity, and is too complex to describe here. Instead, we dedicate three chapters in the part IV of this work to approach the problem in an incremental and efficient way.

4.4.2 Peer Churn

In peer-to-peer video streaming, a client selects one or more uploading peers that leverage their upstream connection to forward the video content. While many algorithms have been proposed for individual video source, supporting a P2PTV service with multiple TV channels brings in additional challenges. A good performance of the P2P streaming depends on the peers' ability to maintain the existing uploading connections. Unlike the typical P2P systems, where peers may act



(a) Uploading only the secondary streams. (b) Uploading both primary and secondary streams.

Figure 4.15: The peer participation on the video streams.

as contributors for entire period they are turned on, in P2PTV, the viewer activity of browsing through the available channels increases the churn, negatively affecting the overall performance.

In order to counteract the negative effect of the channel-change churn, Wu et al. are the first authors to propose a different approach that decouples the peer downloading from uploading (D. Wu, Liang, et al., 2009). In simple terms, with view-upload decoupling (VUD), the participating peers always download a fixed TV channel (the *upload* channel), which in turn may be uploaded to other neighboring peers, and a changing TV channel (the *view* channel) that is selected by the user. By using their idea, the channel churn is completely eliminated because peers never change their contributing stream. The disadvantage of VUD is the higher bandwidth usage due to peers downloading both the view and upload streams. Later work, by Wang et al., demonstrated that the systems that achieve the best utilization of existing resources, are those where peers have the flexibility to participate in any TV channel (M. Wang et al., 2010).

In our work, we adopt their basic idea while considering the specifics of the IMS network: (i) the resources are reserved, and therefore we must be conservatives in their usage, and (ii) we benefit from the session control algorithm prediction of the user demand. Rather than separate completely the view from uploading, like in VUD, we propose that peers may allocate their bandwidth resources to any TV stream. Because we assume the viewer watches a single TV channel at a time, we distinguish between two types of downloaded streams:

- *primary streams*, corresponding to the current TV channel selected by the user;
- *secondary streams*, which are streams of other TV channels that the UE peer may upload to their neighbors.

The secondary streams use the additional download bandwidth of the UE peers, and they are maintained as long as there exists a demand for them. Demand means either downloading peers connected to those streams, or one or more inactive sessions. Therefore, like in VUD, secondary streams are never affected by the channel change churn and do not generate streaming interruptions. As a general principle, a peer may upload any of the secondary streams at any time. In addition it may upload any of primary streams, but if and only if the peer has sufficient network resources to allocate a new primary stream when the user changes the channel.

The figures 4.15(a) and 4.15(b) illustrate the principle of peer participation with primary and secondary streams. In the example on the left, the UE peer uploads only the secondary

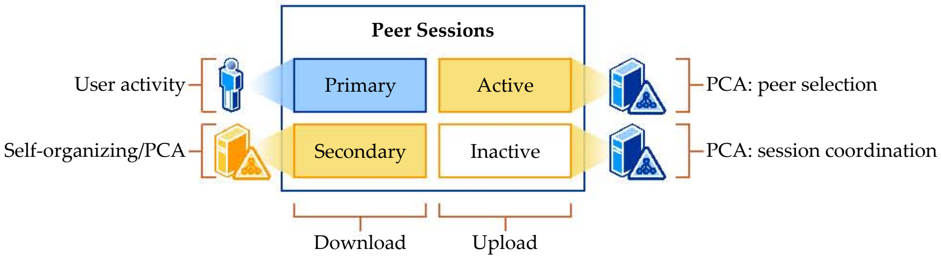


Figure 4.16: The session management at the peer level.

streams, and *generally* it is forbidden to upload primary streams because it does not have sufficient bandwidth to create a new primary stream when changing the channel. By generally, we imply that it is the desired behavior. We shall see later, in the chapter 14, that the operator has the capability of selecting whether the peers should contribute primary streams as well, by accepting a lower delay and higher signaling traffic when the P2P network is low on resources. The example on the right shows a UE peer uploading both types of streams, when it has available download bandwidth.

The figure 4.16 summarizes the management of sessions at the peer level, where we distinguish between the following sessions types: *downloading primary*, *downloading secondary*, *uploading active*, and *uploading inactive*. The primary streams are always determined by the user viewing activity, and they update at every channel change. All other session types are managed automatically by the P2PTV-AS. Give the constraint of the existing network resources, such as the available bandwidth, the AS uses a *peer coordination algorithm*, or PCA, to match the allocation of streams with the existing online peers.

In the part IV, the chapter 14 of this work, we present in greater depth the peer coordination algorithm. In principle, the algorithm is an heuristic that computes an approximate solution to the optimization problem of peer assignment constrained by the existing resources. To this end, the secondary sessions are allocated indirectly by choosing primary streams and these streams change to secondary when the peer changes the TV channel. The active upload sessions are determined through the process of peer selection, and the inactive sessions are computed using the session coordination algorithm. The assignment of these session to the UE peer uses a metric called *resource index*, which considers peer properties such as resource utilization and the available bandwidth.

4.5 Conclusions

The core idea behind the peer-to-peer television (P2PTV) service is that the IMS-enabled user equipments leverage their available bandwidth to support the distribution of the TV channels. Although the architecture of the service, which considers all functional elements of the IMS-based NGN, is a complex one, in our work we focus mainly on those elements that have a significant importance to the service operation: the user equipment (UE), and the P2PTV application server (P2PTV-AS).

The P2PTV-AS is a SIP applications server that intermediates, in a transparent manner using a SIP back-to-back user agent (B2BUA), the establishment of end-to-end media sessions between the UE peers. At the same time, the UEs assigned the network bandwidth, not used for other services, to the P2PTV sessions. This assignment, which requires a corresponding agreement between the telco and the P2PTV provider, harnesses the unused access network capacity and allows individual third-party IPTV content providers to contract the P2PTV distribution, rather than installing and maintaining their own broadcast servers.

In the media plane, we consider the limited resources of the access links. To this end, the TV channels are divided into a number of streams, which increases the bandwidth granularity and robustness to peer failures. We prefer a pull-push approach, where peers only request the first video segment during the P2P session establishment. The pull maintains the playback buffer continuity, when the peers are out-of-sync, while the push limits the P2P signaling during the streaming.

Finally, considering the unique characteristics of a TV service providing multiple channels to the viewer, as opposed to the streaming of a single video, we propose two enhancement methods to tackle the signaling delay and peer churn. The first uses inactive multimedia sessions that enables peers to begin uploading without waiting for the end-to-end establishment of a new session. Because the inactive sessions are effective only if they can accommodate the user demand, the P2PTV-AS uses a session control algorithm to predict the necessary number for the near future.

The second enhancement requires the UE peers to download and upload video streams other than of their current TV channel. This improvement ensures that when the viewer channel the channel, the downstream neighbors are not affected by the change. The assignment of these streams is performed by the P2PTV-AS using a peer coordination algorithm. This considers the demand for different streams and the constraints of the existing network capacity, to compute a heuristic solution.

Chapter 5

Signaling Protocol

5.1 Overview

The control procedures between the user equipments (UE) and the application server (P2PTV-AS) use the SIP protocol, which is supported by the IP Multimedia Core Network Subsystem (IMS). The advantage of the SIP and the IMS is that they guarantee an adequate resource reservation in the access network of the UE to support the video streams. In principle, any end-system supporting the 3GPP SIP profile can be regarded as a compliant UE (3GPP, 2012a), although, for the sake of simplicity, in our work we assume that all UEs are IPTV dedicated devices such as the subscriber's set-top box.

This chapter describes the IMS signaling procedures required by our P2PTV system (Bikfalvi et al., 2009b). To this end, we put a special emphasis on procedures required for normal operation, such as UE connecting to, changing, or disconnecting from a TV stream, as well as signaling of abnormal situations, which include the fast recovery or orphaned UE peers. In our presentation, we do not describe the procedures that are standard for an IMS service and we consider out of the scope of our work. These include the UE registration and deregistration, the P2PTV service configuration such as the service filter criteria stored in the HSS, or more advanced procedures related to policing and charging. These mechanisms are mapped to the IMS control procedures, such that multimedia sessions can be established, modified and released between the parties at the transport plane, that is broadcast servers (BS) and the user equipments. We note that our signaling conforms with the IMS specification, without requiring any changes to the existing standards.

During the access of the P2PTV service, the UE maintains a number of SIP dialogs with the P2PTV-AS.

- A pair of service *subscription* dialogs for notification of AS and UE events.

- A number of originating media sessions, one for each *primary stream* of the current TV channel.

- A number of originating media sessions, one for each *secondary stream* that is contributed by the UE peer.
- A number of terminating media sessions, one for each stream that is *actively* uploaded by the UE peer.
- A number of terminating media sessions, one for each *inactive* or *stand-by* sessions of the UE peer.

The subscription dialogs (Roach, 2002) are established during the initialization of the UE, after the registration to the IMS. One subscription is initiated by the UE, on which the P2PTV-AS transmits to the UE as an XML body a description of the available TV channels, including the corresponding streams for each channel, audio/video format and bandwidth requirements. This information is critical for the UE to prepare the session requests during the connection to a new TV channel. On the second subscription, initiated by the AS following the UE subscription, the UE notifies the P2PTV-AS of its contribution status, such as whether it has sufficient bandwidth to act as an uploader, and which are the supported streams. In addition to this mandatory notifications, the service subscription dialog may be used to transmit optional data such as the TV guide and other service features. In the appendix sections A.2.4 and A.3.5, we present the setup of both subscription dialogs.

Typically, only the service subscriptions and primary stream dialogs are maintained active while the UE is online. All other sessions are used only during uploading requests received from the P2PTV-AS. The upload sessions, either active or inactive, are initiated by the AS and terminate at the UE, while secondary stream sessions are former primary stream sessions for which the UE peer is still an active contributor.

5.2 Media Session Management

All media sessions are stream-based. When a UE tunes in to a selected TV channel, it uses the service information, received on the UE subscription dialog, to initiate a number of multimedia sessions for the corresponding TV channel streams to the media function (MF), another UE or the broadcast server (BS). Typically, download sessions originate at the UE, while upload sessions originate at the P2PTV-AS. However, under certain circumstances, download sessions for the secondary streams may also be initiated by the P2PTV-AS when the number of contributing peers for a given stream is too small, yet the peers collectively has a significant amount of unused network resources. The chapter 14 explains in further detail the peer assignment and resource utilization.

The lifetime of a media sessions has three stages: *initiation* or *establishment*, *updates*, and *termination* or *release*. Our design uses session updates to modify any of the MF parties involved in video streaming and/or the video stream, while maintaining the SIP dialog. Because all TV streams have the same bandwidth and QoS requirements, it is not necessary to perform a recommitment of network resources at the PCRF/PCEF or RACS during a session update. This design approach is possible because all session SIP dialogs are between the user equipments and the P2PTV-AS.

Peer-to-peer media sessions established through the P2PTV-AS may differ depending on whether the AS must initiate a new dialog to the uploading UE, or the dialog already exists. The core idea behind the IMS implementation of the P2PTV system is to use uploading peers that already have SIP sessions established in an inactive state, with reserved resources in the access network but no media. Therefore, to distinguish our enhancements, in this section, we introduce the management of media session with peers that have no inactive upload sessions.

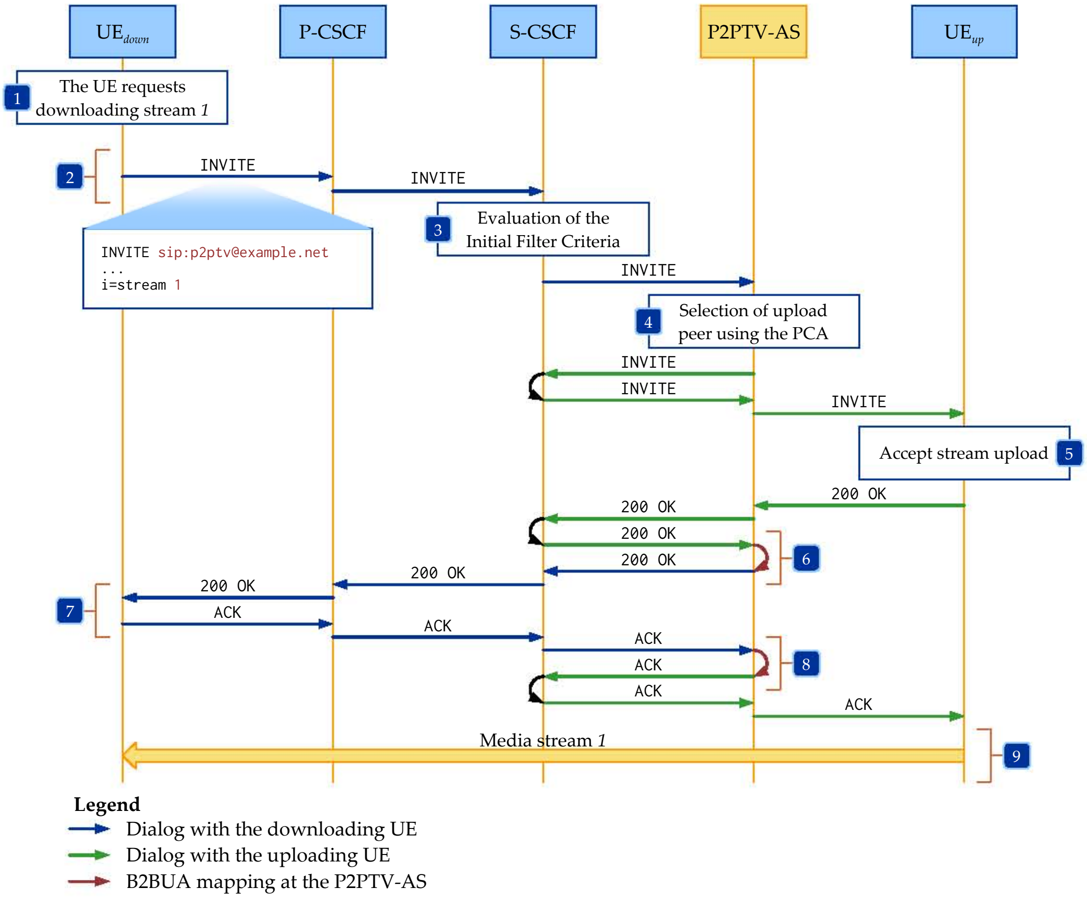

UE_down
P-CSCF
S-CSCF
P2PTV-AS
UE_up
1 The UE requests downloading stream 1
2
INVITE
INVITE
INVITE sip:p2ptv@example.net
...
i=stream 1
3 Evaluation of the Initial Filter Criteria
INVITE
4 Selection of upload peer using the PCA
INVITE
INVITE
INVITE
Accept stream upload 5
200 OK
200 OK
200 OK
200 OK
6
200 OK
200 OK
7
ACK
ACK
ACK
ACK
ACK
ACK
8
Media stream 1
9
Legend
Dialog with the downloading UE
Dialog with the uploading UE
B2BUA mapping at the P2PTV-AS

Figure 5.1: The session establishment procedure when the streams are uploaded by peers.

5.2.1 Session Establishment

5.2.1.1 Peer-to-Peer Session

During a session establishment, the access network is configured to provide the corresponding quality-of-service network resources, such as bandwidth allocation and QoS class. The figure 5.1 illustrates the simplified session establishment procedure for a selected TV stream, when the corresponding media function is another UE peer. We assume that at the beginning of the procedure both UEs have a dedicated access network bearer for SIP signaling and have performed the IMS registration procedure (3GPP, 2012c). In the interest of clarity, we omit the signaling steps that are required for mandatory preconditions, which we present in the appendix A.

The procedure begins when the UE generates a set of SIP INVITE requests, one for each stream of the TV channel, addressed to the public service identity of the IPTV service (e.g. sip:p2ptv@example.net) and contains an SDP payload describing the multimedia session including the requested TV channel/stream, media type and format, the IP address and port where the video will be received, transport protocol, bandwidth and direction (3GPP, 2012g).

The INVITE request is sent to the IMS core (step 2), where it is routed towards the S-CSCF assigned to the end user. The S-CSCF performs the procedure for service provisioning by checking the content of the INVITE request against the set of initial filter criteria that are associated with the public user identity of the user (step 3). These filter criteria are contained in the user profile downloaded from the HSS in the S-CSCF during the registration procedure. If the user has the rights to access the P2PTV service, the S-CSCF will forward the request to the P2PTV-AS.

The P2PTV-AS verifies the user has rights to access the TV stream, based on the subscription

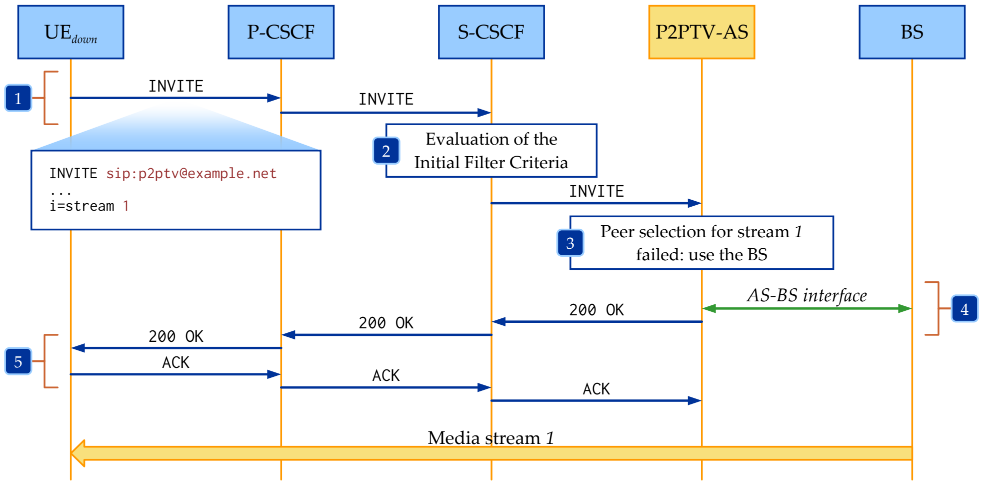


Figure 5.2: The session establishment procedure when no uploading peers are found.

information, and if true, uses the *peer coordination algorithm* or PCA to find the identity of the UE peer that can upload the requested stream (step 4). The detailed operation of the PCA is described, as part of application server, in chapter 14. If a peer is found, the P2PTV-AS performs the functions of a SIP back-to-back user agent (B2BUA) by establishing a new originating multimedia session with the UE of the uploading peer (Rosenberg et al., 2002). The SDP payload of the new `INVITE` request includes the same session-related information as provided by the initiator UE.

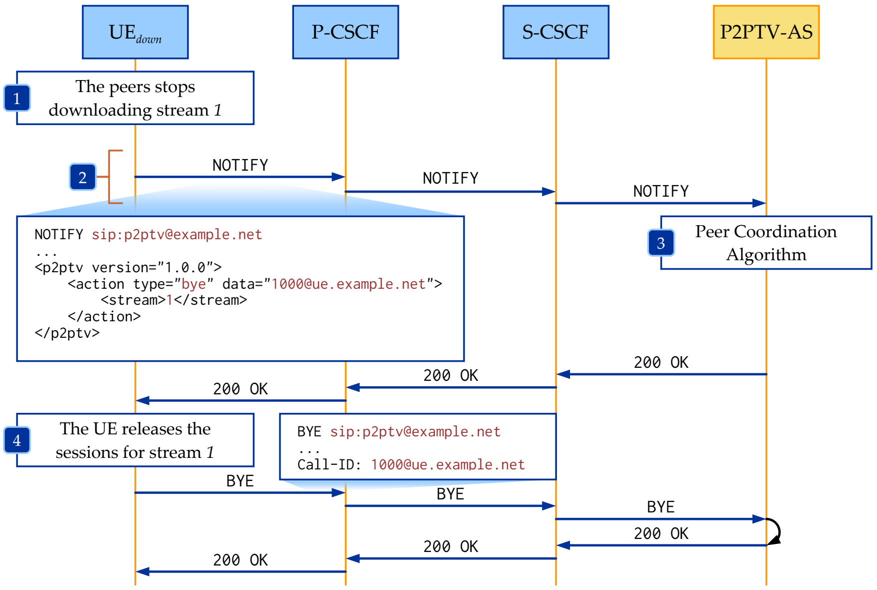
The uploading UE peer accepts the incoming sessions automatically (the whole process is invisible to the user), providing that it meets the uploading requirements, and replies to the P2PTV-AS with a `200 OK` response (step 5). The `200 OK` includes an SDP payload containing session related information, such as the IP address and port where the uploading peer receives RTCP reports, if RTP is used for the transport of media, and the video format that will be used for the TV stream. Upon receiving the response, the IPTV-AS will generate a new `200 OK` response to the initiating UE, which includes the SDP provided by the uploading peer (step 6). Finally, the downloading UE completes the signaling procedure confirming the reception of the `200 OK` response with a `ACK` message (step 7), replicated by the P2PTV-AS for the uploading peer (step 8).

During the session establishment, the QoS provisioning is handled by the either the PCRF/PCEF or RACS, based on the information received from the P-CSCF (3GPP, 2012h) and UE or the access network, depending on whether we deploy the 3GPP (3GPP, 2012g) or TISPAN (TISPAN, 2011b) implementation of the IMS. The resource allocation is done for each upload/-download session in two stages: during the initial `INVITE` the P-CSCF contacts the PCRF/PCEF or RACS to perform the resource reservation (according to the parameters in the SDP offer), and the reserved resources are committed by the PCRF/PCEF or RACS when receiving the `200 OK` response (according to the parameters in the SDP answer).

The UE uses the AS subscription dialog to notify the P2PTV-AS of changes in the uplink/-downlink resources and the affinity of the UE peer to participate as an uploader. In the chapter 14, we provide a detailed description of the interaction between the UE and the P2PTV-AS in terms of advertising the UE capabilities. For now, it suffices to say that the UE sends new notifications whenever the state information changes.

5.2.1.2 Peer-to-Server Session

If during the peer selection process, the peer coordination algorithm of the P2PTV-AS cannot find any third party peer capable of uploading the TV stream (for instance, when there are no other users watching the same channel), it will connect the new UE to a preset broadcast server (BS)


UE$_{down}$
P-CSCF
S-CSCF
P2PTV-AS
1
The peers stops downloading stream 1
2
NOTIFY
NOTIFY
NOTIFY
3
Peer Coordination Algorithm
200 OK
200 OK
200 OK
4
The UE releases the sessions for stream 1
BYE
BYE
BYE
200 OK
200 OK
200 OK

```
NOTIFY sip:p2ptv@example.net
...
<p2ptv version="1.0.0">
    <action type="bye" data="1000@ue.example.net">
        <stream>1</stream>
    </action>
</p2ptv>
```

```
BYE sip:p2ptv@example.net
...
Call-ID: 1000@ue.example.net
```

Figure 5.3: The session release procedure.

delivering that TV channel. The signaling that are involved, illustrated in the figure 5.2, are similar to the peer-to-peer scenario, except that the communication between the P2PTV-AS and the BS, shown in green, may or may not involve the core IMS.

Defining this interface is outside the scope of our present work. However, it suffices to say that a number of options are available to implement the AS-BS communication, depending on whether the BS conforms with the TISPAN IPTV media function specifications and whether the agreement between the IMS operator and the IPTV provider requires the signaling to be routed through the S-CSCF. One of these options is using the $y2$ reference point between the core IMS and the media control function (MCF) of the broadcast server, as illustrated by the figure 3.7 (TISPAN, 2011a). Alternatively, the P2PTV-AS may generate requests to the broadcast server directly, using the PSI of the P2PTV service as originating party, without involving the S-CSCF and therefore bypassing $y2$ (3GPP, 2012c). For the last approach, the P2PTV-AS is required to implement the network domain security and charging functions (3GPP, 2011a, 2011c).

5.2.2 Session Termination

When the UE stops downloading a TV channel stream[1], the UE must release the corresponding download sessions. Because each session represents a branch in the streaming tree, it is essential to maintain the peer participation while downstream peers connect to a different uploading UE or the broadcast server. To this end, before the graceful termination of any download session, the UE notifies the P2PTV-AS on the AS subscription dialog that it plans to stop downloading the selected stream. The XML body of the `NOTIFY` request includes the intended action or actions and the affected streams (step 1 in the figure 5.3)[2].

Upon receiving the request, the P2PTV-AS uses the peer coordination algorithm to take an appropriate action to the session termination (step 3). The chapter 14 describes in further detail the behavior of the AS when the UE ends a session. Following the execution of the appropriate PCA function, the P2PTV-AS answers back to the `NOTIFY` request with a `200 OK` response, allowing the peer to release the multimedia sessions (step 4). Failures, such as network or power outages,

[1] In our current design, a UE stops downloading a stream when being turned off by the user, or it must reuse the downstream bandwidth for a different stream.

[2] The section A.5, from the appendix A, presents in further detail the XML schema used for P2PTV service notifications.

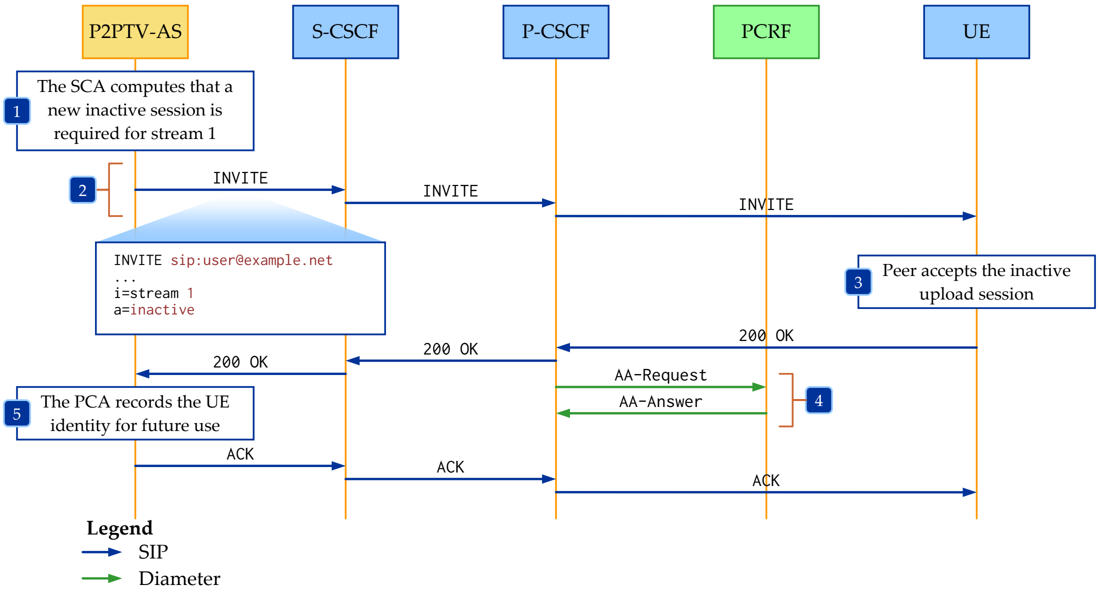

P2PTV-AS
S-CSCF
P-CSCF
PCRF
UE
1
The SCA computes that a new inactive session is required for stream 1
2
INVITE
INVITE
INVITE
INVITE sip:user@example.net
...
i=stream 1
a=inactive
3
Peer accepts the inactive upload session
200 OK
200 OK
200 OK
AA-Request
AA-Answer
4
5
The PCA records the UE identity for future use
ACK
ACK
ACK
Legend
SIP
Diameter

Figure 5.4: The establishment of an inactive upload session.

are detected by the other UE peers and/or the P2PTV-AS and are recovered independently. Failure detection uses SIP timers and the streaming protocol to sense the loss of video data at upstream/downstream media functions. Failures are reported to the P2PTV-AS on the AS subscription dialog.

5.3 Fast Session Mechanisms

The fast session mechanisms are an enhancement over the normal end-to-end SIP signaling. The session establishment is an time-expensive process, mainly due to the signaling involved in reserving and committing the required QoS resources. While the downloading sessions for the primary channel streams are established once at the initialization of the UE, the uploading sessions follow the dynamics of peer participation and selection. For example, in the process of changing the TV channel, although a UE reuses the same download sessions, it requires new uploading sessions at all new corresponding peers.

5.3.1 Inactive Sessions

In order to accommodate this unbalanced situation in the peer-to-peer scenario, we proposed to use UE peers that already have established a number of inactive upload sessions (Bikfalvi et al., 2009b). We call these stand-by peers as *fosters*, by being ready to adopt future orphaned sessions. The number of inactive sessions is per-channel property (all streams for a channel require an equal number of inactive sessions) and it is computed at the P2PTV-AS. To this end, the AS software uses the *session coordination algorithm* or SCA to determine the number of inactive sessions that are sufficient to serve the existing download demand.

Given the complex nature of the TV user activity, computing the optimal number of inactive sessions is a complex matter. Ideally, the number of sessions should be large enough to server any request, but small enough to prevent the over-reservation (and waste) of network bandwidth. While in theory this is an optimization problem, applying it to our current system presents important challenges. In the chapters 11, 12 and 13, we present three different control techniques that, when working together, solve the inactive session allocation problem in a way that is both

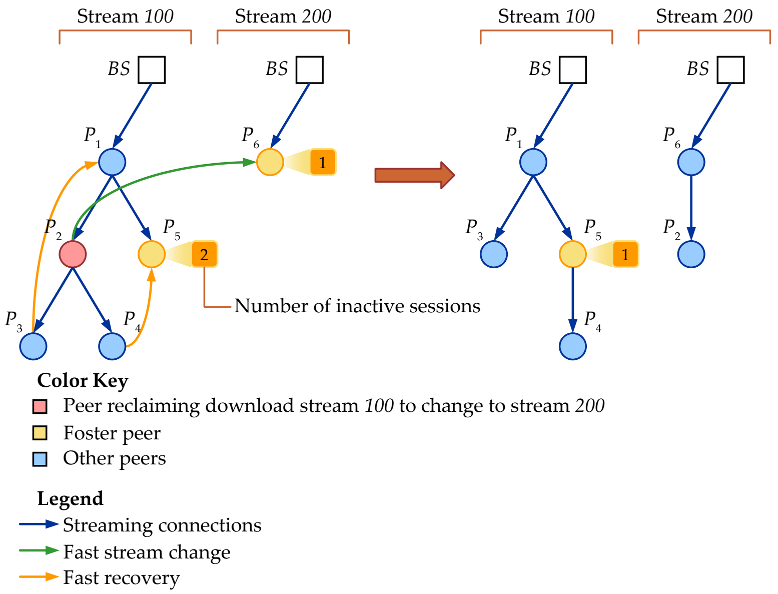


Figure 5.5: Fast stream change and fast recovery.

balanced, between the request blocking ratio and the resource utilization, and adaptive to the user activity.

The figure 5.4 illustrates the establishment of an inactive upload session. The P2PTV-AS uses the session control algorithm to compute at regular intervals[3] the number of inactive sessions required for each TV channel stream (step 1). Establishing an inactive sessions follows the same procedure as an active session, described in the previous sections, except that the AS marks the session with the inactive attribute in the SDP body of the initial `INVITE` (step 2). This attribute tells the uploading UE to setup a transport bearer for the stream but without beginning the streaming of video data (step 3). When receiving the SDP answer from the UE, the P-CSCF authorizes the QoS reservation at the PCRF (step 4). Finally, the P2PTV-AS uses the peer coordination function to record the UE identity and the corresponding stream for future use.

5.3.2 Fast Stream Change

The fast stream change is an enhanced signaling procedure that the user equipments use during channel switching. It relies on the use of foster peers and inactive upload sessions to improve the system response time and the service quality perceived by the viewer. Más et al. list a time of less than 500 milliseconds as an adequate channel change delay in multicast-based IMS-TV service (Más et al., 2008), which sets a very high bar for the P2PTV approach if it were to present itself as a serious alternative.

The figure 5.6 illustrates a fast stream change scenario: a UE peer, P_2, replaces stream 100 with stream 200 as part of a channel change. During the signaling procedure, the PCA at the P2PTV-AS selects foster peer P_6, having one inactive upload session for stream 200, as uploading peer. To emphasize the benefit of foster peers in the same figure, we assume that the departure of peer P_2 from stream 100 requires the recovery of two orphaned downstream sessions to P_3 and P_4[4].

The figure 5.6 shows the corresponding signaling procedure. In the *fast stream change*, after

[3] Every 10 seconds in our current design.

[4] Orphaning downstream peers during a channel change is an atypical situation, since UE peers are required to maintain sufficient reserve bandwidth to accommodate uploads for the primary streams (secondary streams are not affected by channel changes). However, the service provider may customize the reserve threshold to a lower level, especially when they have only a handful of HDTV channels requiring more streams. In addition, session orphaning always happens when the STB is turned off.

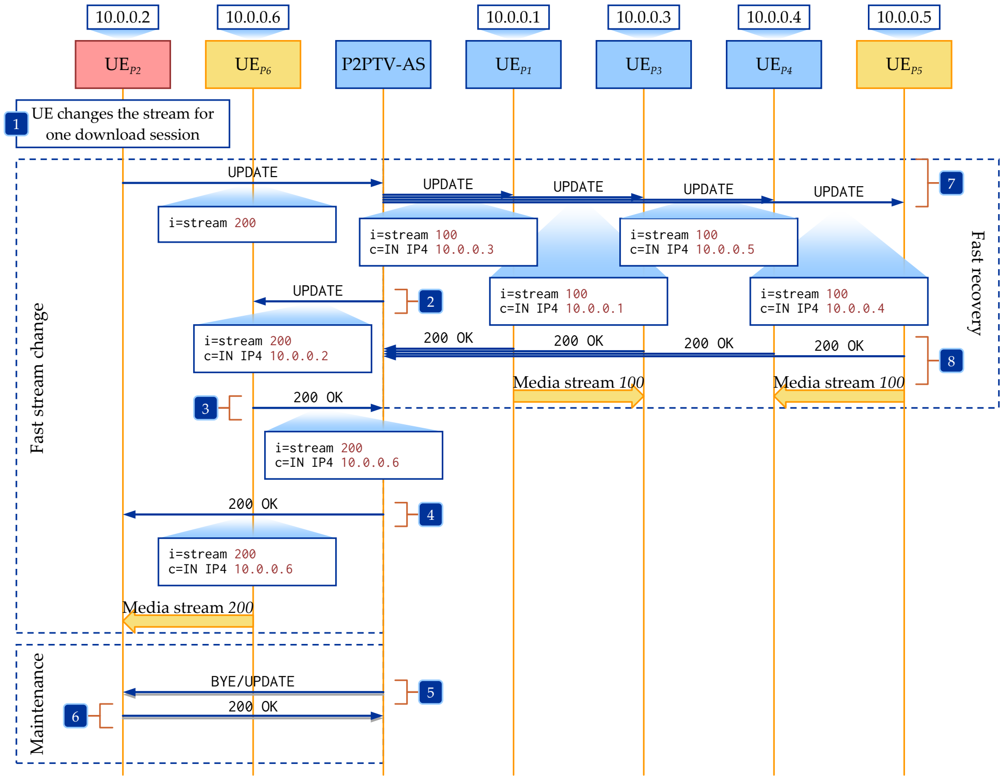


Figure 5.6: Session signaling for fast stream change and fast recovery.

the departing UE initiates the session update by attaching a different stream number to SDP session information (step 1), the P2PTV-AS selects foster peer P_6 as uploading UE. The AS creates a mapping between the two sessions in the B2BUA and activates the inactive session by sending an UPDATE request to the foster peer with the connection information of P_2. The 200 OK response from P_6 completes the fast stream change (steps 3–4). It is worth noting, that the signaling procedure does not involve any other IMS or access entities such as the policy control functions or the RACS. Following the stream change, both the SCA and PCA update the assignment of the inactive session.

Following a stream change, the P2PTV-AS executes a maintenance procedure to release or update (steps 5–6). The AS has the following options: (i) to release the upload sessions if further participation of the selected peer is no longer needed; (ii) change the upload sessions to inactive, either for the new stream or for any other stream that is being downloaded by the peer; (iii) use the upload sessions to re-accommodate other peers from the system. The last option is a streaming optimization procedure that we discuss further in the next sections. Recycling existing upload sessions takes advantage of network resources that are allocated within the bearer part of the access network.

5.3.3 Fast Stream Recovery

The recovery of orphaned sessions, when the user turns off its device or when the peer changes a download stream, may also benefit from the inactive upload sessions. The reconstruction of the streaming tree requires the immediate attention of the P2PTV-AS, since a high delay in reestablishing the broken sessions may lead to service quality degradation to a large number of subscribers. In the figure 5.5, we illustrate the session recovery when the peer P_2 changes a download session to a different stream.

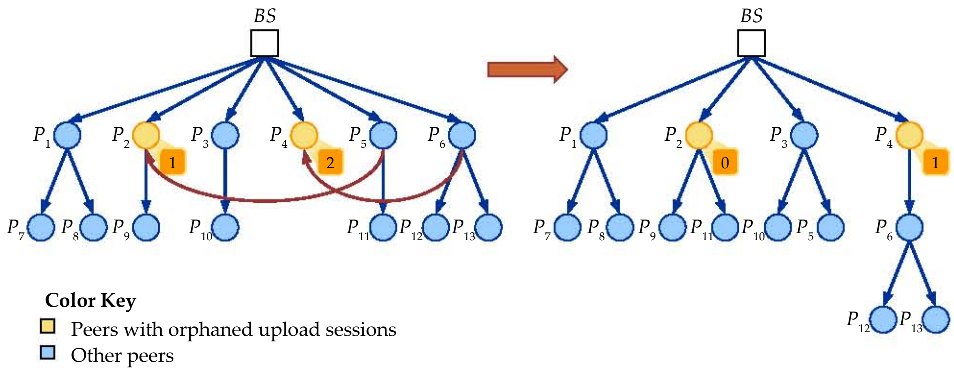


Figure 5.7: Streaming optimization to reduce the upload from the media server.

The fast stream recovery uses the following approach: when the uploading entity of the *departing*[5] peer is another peer, its upload session is mapped to one of the downstream peers. The remaining orphaned sessions are connected to one or more foster peers. In our example, the orphaned P_3 connected to the upstream peer P_1, while P_4 used one inactive session of P_5.

The figure 5.6 shows the corresponding signaling flow. As in the case of fast stream change, the procedure takes only two messages, an `UPDATE` (step 7) and a `200 OK` (step 8), making possible the re-establishment of the stream in a single round-trip-time. Typically, all orphaned sessions are recovered in parallel, with the video stream resumed after the `200 OK` message. Both fast stream change and fast stream recovery only work if there are a sufficient number of inactive upload session to accommodate the demand. Otherwise, the P2PTV-AS shall use either the broadcast server or normal UE peers, depending on the policy of the service provider.

5.3.4 Streaming Optimization

The streaming tree structure is determined by users tuning in (turning on their devices), changing channels and leaving (turning off their devices). Due to the randomness of users performing these actions, it is possible to obtain streaming trees that do not meet the service providerŠs goals in terms of users connected to the broadcast, usage of available resources and video quality. Because the P2PTV-AS maintains a global view of the system, it can adjust the structure of the streaming tree by issuing session updates. We call this adjustment streaming tree optimization, and it represents a generic procedure that modifies the tree in order to optimize a small set of parameters as defined by the policy of the service provider.

Streaming optimization is a procedure that benefits from the recycling of established but orphaned upload sessions. Typically, an upload session is orphaned, when the downstream UE turns off or changes to a different stream. More rarely, orphaning happens when a peer forcibly changes a download stream requiring downstream UEs to recover to different uploading peers. When upload orphaning occurs, the P2PTV-AS has two options: (i) change the upload session to inactive, making the peer a foster, or; (ii) using the upload sessions to modify the structure of the streaming tree. Since the appropriate action is decided by the peer coordination algorithm, the chapter 14 covers this topic in further detail, especially when and how the decision is taken.

From the signaling perspective, the figure 5.7 illustrates a streaming optimization scenario, designed to reduce the number of streams uploaded by the broadcast server. In this particular circumstance, two peers, P_2 and P_4, receiving the same stream have three orphaned sessions from previous uploading of the same or different stream[6]. To decrease congestion on the broadcast server, the P2PTV-AS selects two UEs, located on a different streaming tree branch, which are

[5] In this context, by departing peer we understand a peer *leaving* a stream, and not necessarily turning off.

[6] The recycling of upload sessions is limited to streams that share the same resource requirements.

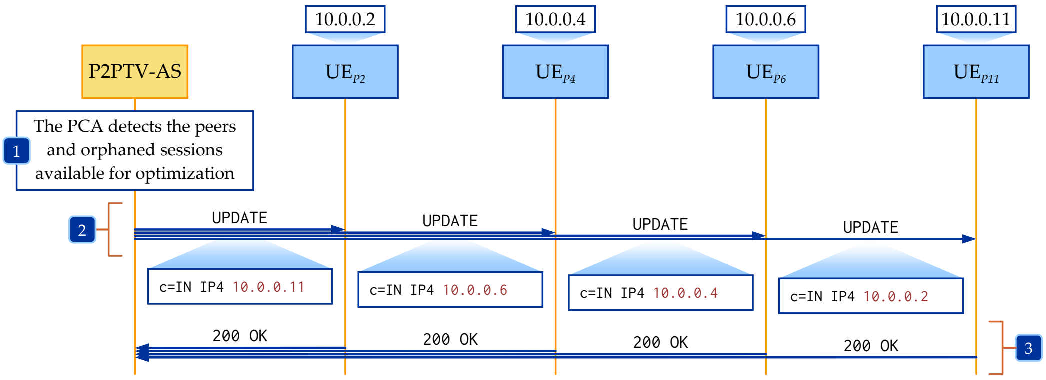


Figure 5.8: Streaming optimization to reduce the upload from the media server.

transferred to the orphaned sessions. The figure 5.8 shows the corresponding signaling messages, consisting of an UPDATE issued by the P2PTV-AS with the new SDP connection settings(step 2), and an 200 OK response returned by the UE peers.

5.4 Conclusions

During the access of the P2PTV system, all UE peers maintain a number of SIP dialogs with the P2PTV application server. On the first dialog, setup during the initialization, the UE maintains a subscription to the P2PTV service that allows two-way notifications to and from the P2PTV-AS. These notifications inform the UE on the settings of the service, including the technical characteristics of the TV channels, and allow the UE to report state information. The remaining SIP dialogs are dedicated exclusively to downstream and upstream media sessions. They are initiated either by the UE, typical for download, or by the P2PTV-AS, for upload. All signaling dialogs are between the UE and the P2PTV-AS, and the peer-to-peer communication is always relayed through the AS back-to-back user agent, making the P2P completely transparent to the devices.

In order to avoid the expensive delay of establishing new QoS-enabled transport bearers during time-sensitive events, such as a channel change or an uploading peer departure, the AS uses the inactive upload sessions. These correspond to existing transport bearers, and require fewer signaling steps. To this end, we describe three enhanced signaling procedures, fast stream change, fast stream recovery and streaming optimization, which are significantly lighter than the standard counterparts.

Chapter 6

Support for Mobility

6.1 Overview

The P2PTV service works with any type of IMS-based terminals and is not limited to residential networks. However, support for wireless devices must consider that the access bandwidth is a scarce resource and the uplink should not be used by the UE in order to minimize costs and improve the resource usage. This constraint is addressed at the P2PTV-AS, by guaranteeing that a mobile UE never assumes the role of an uploading peer for any streaming tree.

In addition to bandwidth limitation, a mobile UE may change its IP address while receiving a stream during roaming between two access networks. Handovers are classified as *soft* or *make-before-break*, where the UE establishes a connection to the new network before disconnecting from the old one; and *hard* or *break-before-make*, where connectivity is lost during the network change. While some existing studies addressed the soft handovers for IPTV streams in WLAN scenarios (Cunningham et al., 2009), in this chapter we address the handover issue from a more general perspective, valid for any type of mobile device. Toward this end, we conclude that a proactive approach is necessary for the handling of inter-network handovers. Our solution, called proactive context transfer service, incorporates the existing IEEE 802.21 technology in order to minimize the handover delay (Vidal et al., 2010).

The objective of the handover design is to minimize the loss of streaming data for the IPTV client. We identify two possible approaches:

- introducing enhanced buffering techniques at the uploading and downloading UE peers, to compensate for connectivity loss delay, and;
- reducing the handover delay of the mobile UE.

While video data buffering is a simple and effective solution to maintain streaming and playback continuity, it is always limited by the perceived drop in quality during the initial buffering wait or when an interruption occurs. Therefore, our work insists on the second option of minimizing the handover delay.

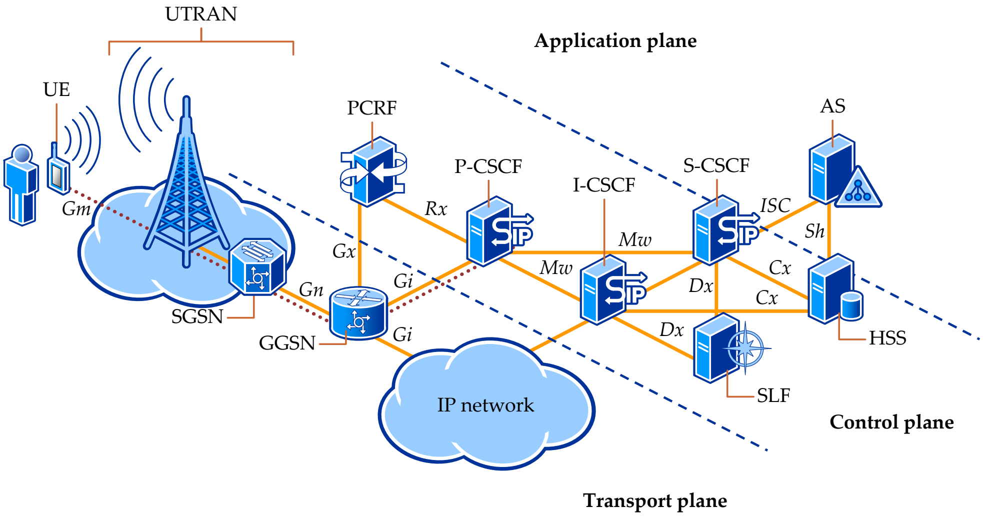


Figure 6.1: The architecture of the IP Multimedia Subsystem in a UMTS mobility scenario.

6.2 Existing Alternatives for UE Mobility

The effort to improve mobility management in IP networks includes the work of the standardization bodies such as the Internet Engineering Task Force (IETF), who have developed solutions for mobility support in IPv4 (Perkins, 2002) and IPv6 (Johnson et al., 2004), for improved IP handover performance (Koodli, 2009; Soliman et al., 2008), and for supporting the movement of networks as a whole (Devarapalli et al., 2005). However, macro-mobility (changing network and IP address) in IMS-based networks is very difficult to achieve with existing standards (Heine & Sagkob, 2003), although this type of mobility is expected to become very common. Chiba et al. and Reiner et al. have analyzed the challenges of integrating mobility, IMS, and control of the access network for WLAN and CDMA2000 and GPRS, respectively (Chiba et al., 2008; Renier et al., 2007). This section describes three existing handover mechanisms for IMS-based services, particularized to our P2PTV service. We assume that the user equipment (UE) uses a UMTS access network, although the mobility procedures are applicable to any IMS-based network and access technology with the exception that the resource reservation in the access network would be different.

The figure 6.1 illustrates the basic architecture of the P2P-IMS-TV system in an UMTS mobility scenario. The UE is connected via the Terrestrial Radio Access Network (UTRAN) and the UMTS packet domain, the latter including the Serving GPRS Support Node (SGSN), which links the radio access network with the packet core network, and the Gateway GPRS Support Node (GGSN), which connects to external networks and provides the UE with IP-level connectivity using PDP contexts. A Packet Data Protocol (PDP) context is a QoS-enabled logical connection that supports the exchange of IP packets between the UE and the GGSN. In the case of GPRS, the GGSN is responsible for the enforcement of the resource reservation, being connected to the PCRF via the *Gx* interface.

In a general mobility scenario, illustrated in the figure 6.2, the UE can be located either in the home or in the visited network. The home network maintains the user subscription data and provides services. We assume that the UE moves from an *old* (or *current*) to a *new* (or *target*) network, where both the old and the new network can be the home or a visited network. The UE entry-point for the IMS, the P-CSCF, can be located either in the home or in the visited network, depending on how its discovery is configured in the UE and in the visited network.

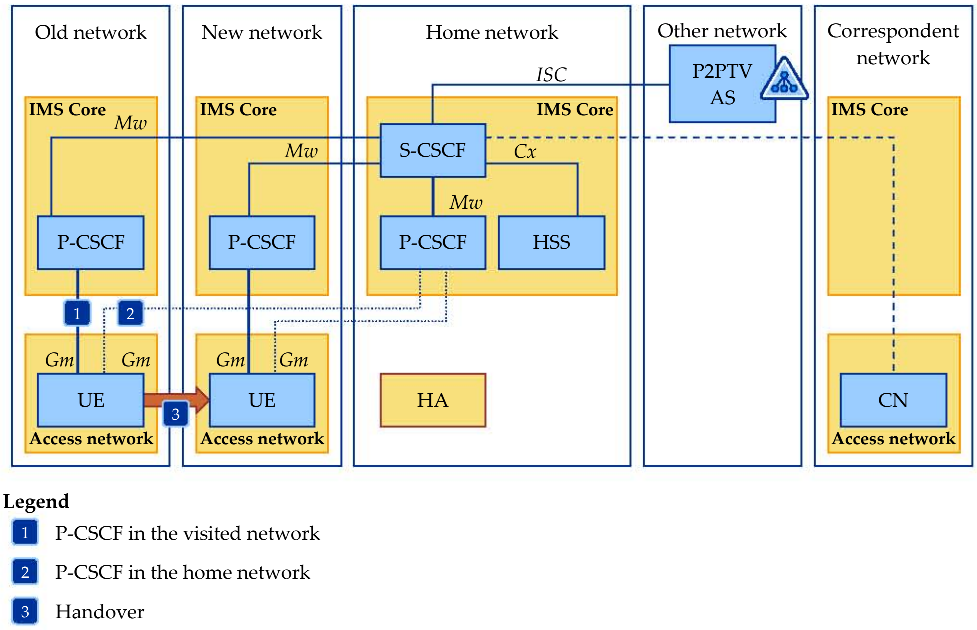


Figure 6.2: The schematic of a general mobility scenario.

6.2.1 SIP Mobility

The SIP mobility uses the existing IMS infrastructure and specification to restore the multimedia session after the user moves to a new network. For the sake of simplicity, we study only the case where the P-CSCF is located in the visited network and a handover implies a change of the P-CSCF assigned to the UE. The situation where the P-CSCF is located in the home network can be derived as a particular case. The figure 6.3 illustrates the IMS signaling flow for the UE handover. When the UE detects that it is about to lose connectivity to the current network, it sends a stream *pause* notification via the P2PTV-AS to the correspondent peers (step 1), which will suspend the upload and will begin buffering a limited duration of each video stream (step 2).

For a UMTS access technology, the UE peer begins the handover process by deleting all bearer PDP contexts, detaching from the old network, attaching to the new network and establishing a primary PDP context for IMS signaling (step 3). With the activation of a new PDP context, the UE obtains a new IP address and discovers a P-CSCF in the new network. Subsequently, the UE initiates a regular IMS re-registration procedure, to inform the S-CSCF of the new contact address and P-CSCF address (step 4) (3GPP, 2012a). Following a successful re-registration, the UE modifies the existing P2PTV sessions by issuing a re-INVITE message via the new P-CSCF, updating the contact URI of the UE, the URI of the new P-CSCF and the IP address and port(s) where the media streams will be delivered (step 5). Simultaneously, the UE performs a resource reservation by establishing a secondary PDP context (step 6).

After the secondary PDP context is activated, the UE sends an UPDATE request, via the P2PTV-AS, to the corresponding peers (CP) with the new IP address and port(s) (step 7). Finally, when all stream sessions have been updates, the UE *resumes* the streaming with a notification sent to the corresponding peers. For the mobility scenario to work, the IPTV application at the UE must execute the signaling procedures described here and preserve the state of the existing SIP sessions during the handover, while network connectivity is lost.

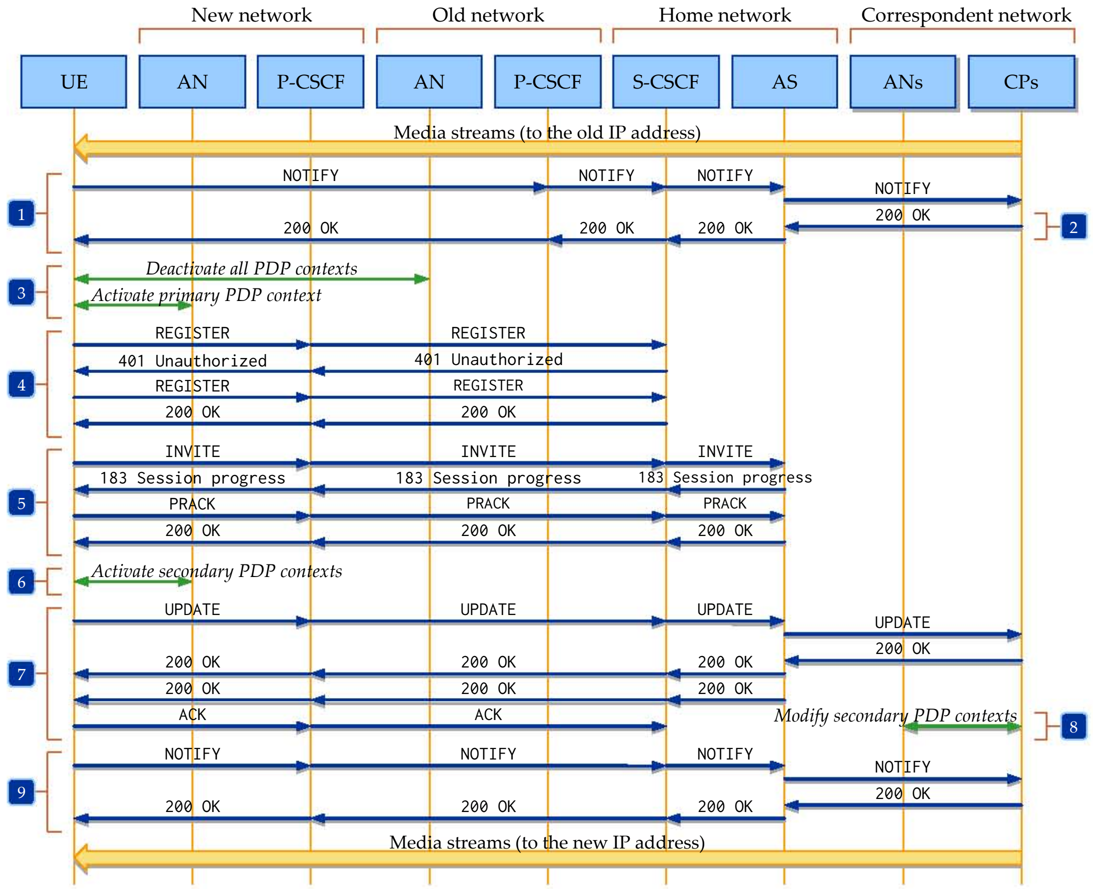


Figure 6.3: SIP mobility with the P-CSCF located in the visited network.

6.2.2 Optimized SIP Mobility

As an improvement over the previous mobility scenarios, Renier et al. and Dutta et al. have proposed an optimization technique that transfers the context information between the old and new P-CSCF (Renier et al., 2007; Dutta et al., 2007). The purpose of the P-CSCF context transfer is to reduce the handover latency by having all the parameters necessary to establish the signaling security associations and the bearer PDP context readily available at the new P-CSCF.

To make the context transfer possible, the IMS architecture has to be modified by changing the P-CSCF and adding a new reference point or interface between two P-CSCFs. The figure 6.4 shows the re-registration procedure that is executed after the activation of the primary PDP context in the new network. The process of context transfer starts with the UE sending an integrity-protected re-registration message to the new P-CSCF, containing the old P-CSCF information (step 1). Because the integrity key is not known to the new P-CSCF at this time, the P-CSCF must defer the verification of the message and UE authenticity until after the context transfer. After receiving the context transfer request, the new P-CSCF contacts the old P-CSCF in order to retrieve the UE context parameters including the encryption keys for the security associations, and the media parameters and filters of the previous sessions (steps 2,3). If the context transfer is successful and the integrity of the re-registration request is verified, the S-CSCF is informed of the UE location change (steps 4,5).

Due to the P-CSCF context transfer, the modification of the streaming session at the UE requires only three message exchanges, as we illustrate in the figure 6.5: an initial re-`INVITE` sent to the P2PTV-AS containing an SDP with the new media parameters at the UE-side, a final `200 OK` response and an `ACK`. Following the reception of the re-`INVITE` request, the new P-CSCF will create

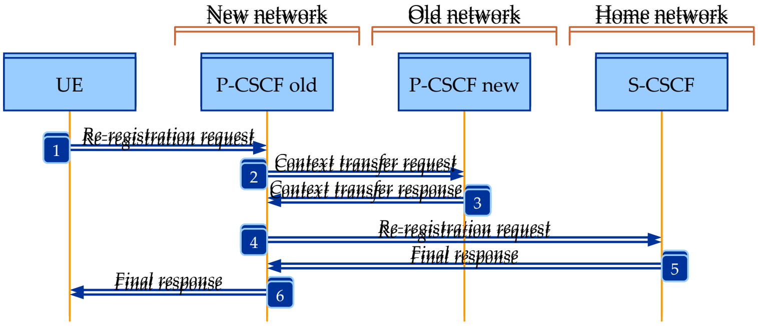


Figure 6.4: Re-registration with P-CSCF context transfer.

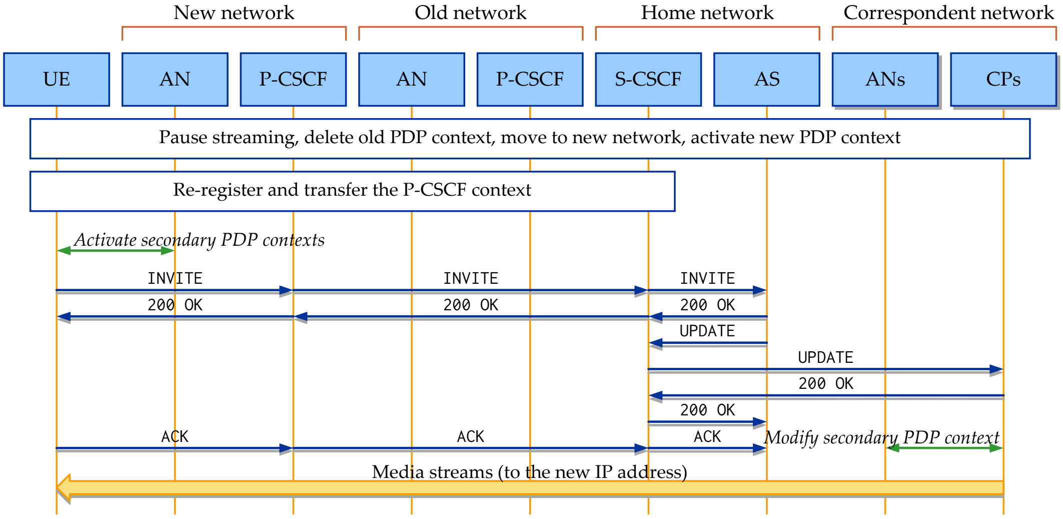

New network
Old network
Home network
Correspondent network
UE
AN
P-CSCF
AN
P-CSCF
S-CSCF
AS
ANs
CPs
Pause streaming, delete old PDP context, move to new network, activate new PDP context
Re-register and transfer the P-CSCF context
Activate secondary PDP contexts
INVITE
INVITE
INVITE
200 OK
200 OK
200 OK
UPDATE
UPDATE
200 OK
200 OK
ACK
ACK
ACK
Modify secondary PDP context
Media streams (to the new IP address)

Figure 6.5: Optimized SIP mobility with P-CSCF context transfer.

a new traffic filter with the media parameters from the transferred context and new UE. At the same time, the UE establishes a secondary PDP context within the new UMTS access network. In parallel, upon receiving the `INVITE` the P2PTV-AS will trigger a session update for the media parameters at the side of the corresponding peers. Finally, a notification to resume the streaming informs the CPs that the handover process is completed.

6.2.3 Mobile IP

Mobile IP (MIP) (Perkins, 2002; Johnson et al., 2004) solves the mobility problem by allowing the UE to maintain network layer connectivity while moving to a visited network. With MIP, the mobile UE has two IP addresses, the Home Address (HoA) and the Care-of Address (CoA). To maintain communications, the mobile UE uses the HoA as a permanent address, making the mobility transparent for applications and the correspondent peer. In order to deliver the IP packets to the UE, the UE also receives a temporary address, the CoA, which is topologically correct in the visited network. A Home Agent (HA) in the home network, the network where the HoA is topologically correct, is responsible for sending and receiving the IP datagrams on behalf of the mobile peer, by using a bidirectional IP in IP tunnel.

The figure 6.6 summarizes the operation of MIP version 6 in tunnel mode. When the UE is roaming in a visited network (step 1), the first step is an authenticated binding procedure (step 2), through which the mobile UE informs the HA of its CoA (step 3). At the end of the binding procedure a MIP tunnel through the HA is established and a communication between the UE and

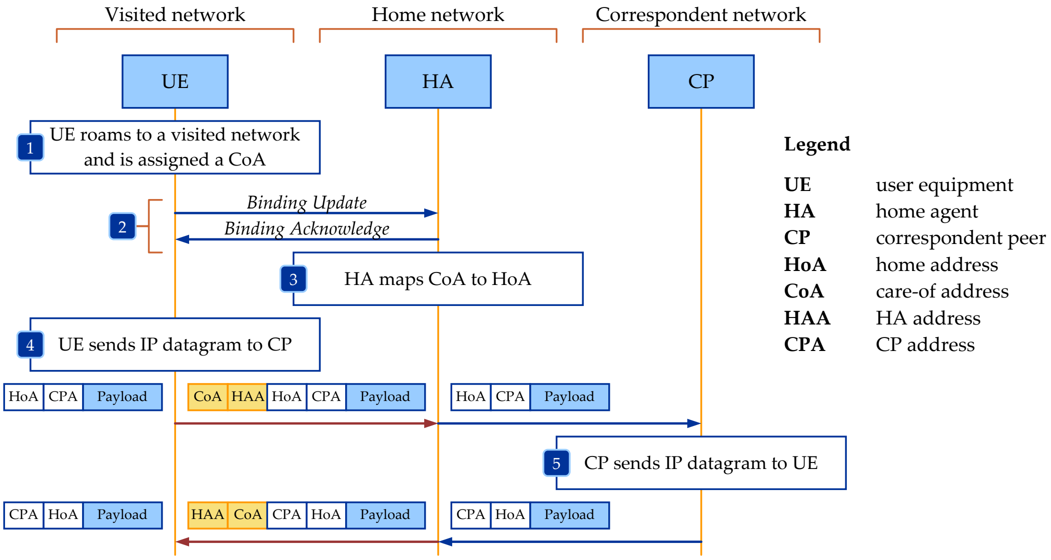


Figure 6.6: Mobile IP version 6.

a CP takes place as follows. The outbound IP datagrams (from the UE to the CP) are tunneled to the HA with the HA Address (HAA) as destination, which in turn will use the topologically correct HoA to forward the packets to the CP (step 4). The inbound IP datagrams, having the HoA as destination and arriving in the home network, are intercepted by the HA and sent through an IP tunnel to the UE using the CoA (step 5).

While MIP offers an attractive solution for seamless transition to mobile scenarios, its integration with the UMTS access technology and with IMS presents several challenges (Renier et al., 2007). The allocation of network resources, including the provision of QoS, involves the creation of appropriate filters in the network gateways (in the GGSN for GPRS). These filters include the IP addresses of the hosts for which the communication is allowed and they are installed during the session establishment according to the Session Description Protocol (SDP) body in the SIP messages (Handley et al., 2006).

Due to the tunneled communication between the UE and the HA, the use of MIP cannot be made transparent to the UE and to the IMS network. On the one hand, in MIP the HoA is the permanent address of the mobile UE, and therefore it should be used by the CP and the IMS core; on the other hand, the access network serving the mobile UE requires that resource reservation and filtering are done using the IP address assigned to the mobile in that access network (i.e. the CoA). As a consequence, using MIP in IMS requires changes in IMS functional entities to make them MIP aware as proposed by Renier et al. (Renier et al., 2007). However, using MIP presents several benefits such as transparency of UE mobility for the corresponding peers and applications, and a simplification of the IMS procedures after mobility. For instance, if the UE does not use the Route Optimization feature of MIPv6, MIP opens the possibility of implementing the buffering at the home agent (Xia et al., 2008).

The implementation of MIP support depends on whether the P-CSCF is located in the home or the visited network. The figure 6.7 illustrates the MIP mobility scenario involving MIP with a home network P-CSCF. The IMS signaling is tunneled with MIP via the HA and the UE does not have to re-register to IMS when moving to a visited network. During the handover, the HA buffers the corresponding streams (steps 2–7). The approach requires a number of modifications in the UE, the IMS core and session description information. These changes allow the network to recognize the mapping between HoA, CoA and HAA, and establish the appropriate media filters in the access network. When the P-CSCF is located in the visited network, the optimal design

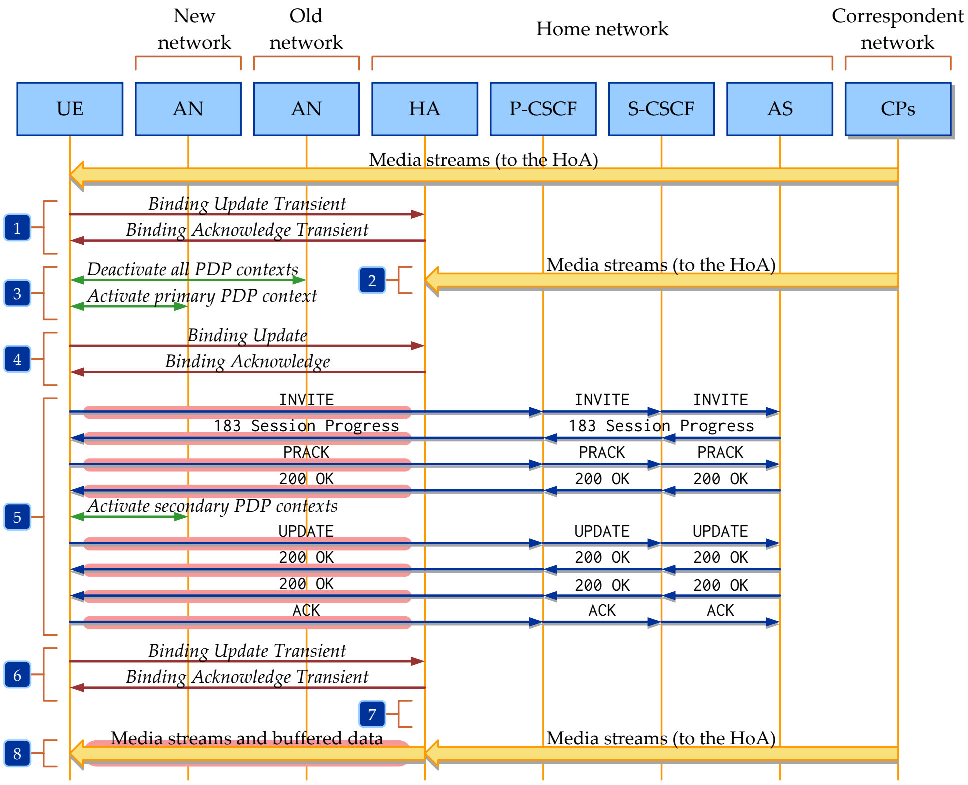

New network
Old network
Home network
Correspondent network
UE
AN
AN
HA
P-CSCF
S-CSCF
AS
CPs
Media streams (to the HoA)
1
Binding Update Transient
Binding Acknowledge Transient
2
Media streams (to the HoA)
3
Deactivate all PDP contexts
Activate primary PDP context
4
Binding Update
Binding Acknowledge
5
INVITE
183 Session Progress
PRACK
200 OK
Activate secondary PDP contexts
UPDATE
200 OK
200 OK
ACK
6
Binding Update Transient
Binding Acknowledge Transient
7
8
Media streams and buffered data
Media streams (to the HoA)

Figure 6.7: The handover procedure with MIPv6 and the P-CSCF located in the home network.

configuration uses MIP only for the media connections. This approach prevents the signaling from being routed from the home agent, back to the P-CSCF in the visited network. The UE uses the CoA for signaling, and therefore, after moving to the new network, it must re-register to the IMS. The media filters remain the same, using the CoA and the HAA.

6.3 Proactive Context Transfer Service

The common property of the existing IMS mobility solutions is that they wait on the user equipment to re-initiate the multimedia sessions after roaming to a new network. These approaches loose invaluable time while the UE is not connected to any network, and then require substantial signaling to inform the IMS and the P2PTV service of the location change. To enhance the roaming response, we propose an active participation on behalf of network, by taking a proactive stance to the mobility procedure. Our approach is based on the transfer of session context between the old and the new network, but unlike previous work, this is done before the UE is connected in the visited network. Toward this end, we resort to the IEEE 802.21 media independent handover procedures to configure the security and QoS state that will be used for the communication between the UE and the new P-CSCF (Gupta, 2008). The IEEE 802.21 introduces a set of primitives enabling the communication between different network entities and the UE.

6.3.1 Service Architecture

The figure 6.8 illustrates the architecture of the Proactive Context Transfer Service or PCTS. The service is provided by a dedicated application server, PCTS-AS. Any user subscribed to this service will connect to a PCTS-AS in its home network, which will be in charge of managing the mobility of the user and initiating the context transfer to a new P-CSCF located in a visited network. The

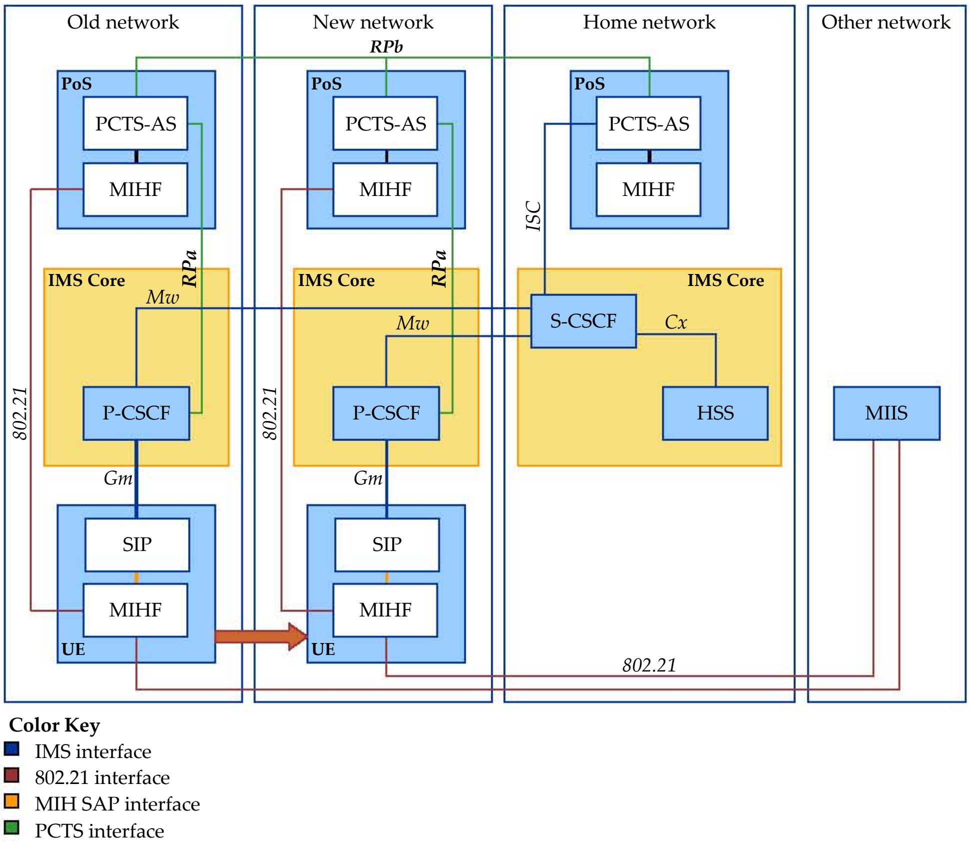


Figure 6.8: The architecture of the Proactive Context Transfer Service (PCTS).

user subscription to the PCTS has a per service granularity, meaning the end user can subscribe to the PCTS for the subset of services that require reduced handover delays, such as IPTV service. Context transfer functionalities may involve other PCTS application servers located in the current and future networks. The PCTS-AS supports an intra-domain SIP interface to the S-CSCF (the *ISC* reference point), a new intra-domain interface to the P-CSCF, denoted generically by *RPa*, and a new inter-domain interface to other PCTS application servers, we denoted generically by *RPb*.

In addition to the standard IMS functionality, both the PCTS application servers and the UEs supporting the PCTS have an additional component, called the Media Independent Handover Function or MIHF, which allows the UEs and the PCTS-AS to exchange handover information via IEEE 802.21. Internally to the PCTS-AS and the UE, the IMS/SIP functions communicate to the MIHF using a MIH service access point, or MIH-SAP, defined by the standard. The PCTS-AS and its corresponding MIH function represent a Point-of-Service, or PoS, managing the transfer of mobility configuration between networks.

6.3.2 Context Transfer Procedure

The proactive context transfer service requires two distinct steps: an *initialization procedure* performed when the UE is turned on, and the actual *context transfer procedure* executed when the mobile UE roams between different networks.

The main objective of the service initialization, illustrated in the figure 6.9, is to subscribe the UE to the notifications of the PCTS, and to inform the PCTS-AS of all multimedia sessions requiring fast handovers by maintaining the AS in the signaling loop[1]. While in the signaling path,

[1] This is achieved by modifying the Initial Filter Criteria at the home S-CSCF.

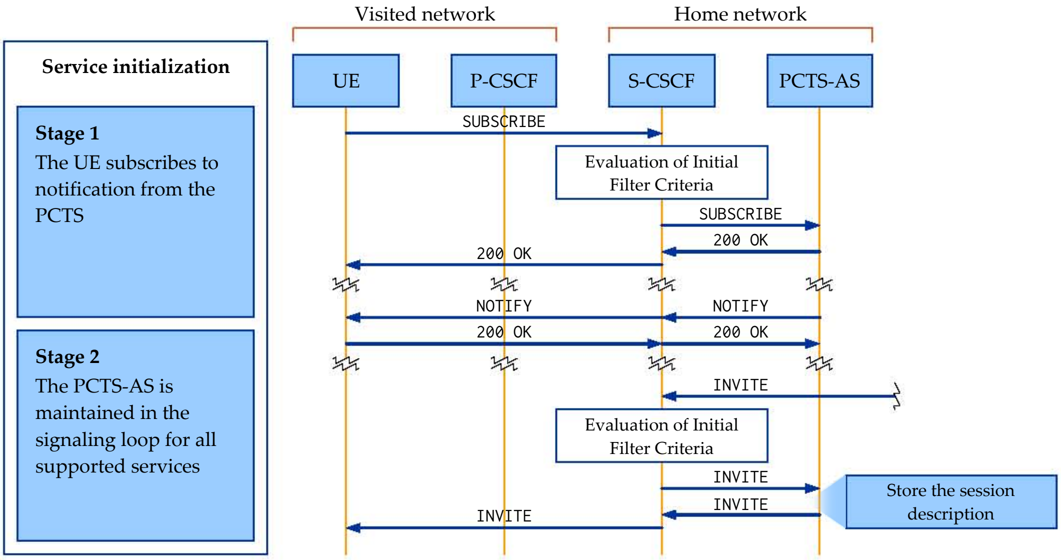

Service initialization
Stage 1
The UE subscribes to notification from the PCTS
Stage 2
The PCTS-AS is maintained in the signaling loop for all supported services
Visited network
Home network
UE
P-CSCF
S-CSCF
PCTS-AS
SUBSCRIBE
Evaluation of Initial Filter Criteria
SUBSCRIBE
200 OK
200 OK
NOTIFY
NOTIFY
200 OK
200 OK
INVITE
Evaluation of Initial Filter Criteria
INVITE
INVITE
INVITE
Store the session description

Figure 6.9: The initialization of the Proactive Context Transfer Service.

the PCTS-AS performs the function of a back-to-back SIP user agent, somewhat similar to that of the P2PTV-AS, by forwarding back to the S-CSCF all incoming SIP requests and responses.

When the UE moves from a current to a new network, it requests the PCTS to transfer the context of all subscribed services to the new P-CSCF. This transfer involves three separated steps:

- obtaining the configuration in the target network, including the new IP address and the URI of the P-CSCF;
- transferring the QoS settings to the new P-CSCF, and;
- reconfiguration of the security context in order to be used in the new network.

The first step uses a MIH Information Service, or MIIS, which stores information regarding existing networks within a geographical area (de la Oliva et al., 2008). Every UE can access the MIIS data via their own MIH function, and determine the P-CSCF and the identifier of different surrounding networks. Based on the information received, the UE triggers a mobile initiated handover by instructing its current PoS to query the PoS of the discovered networks on the availability of resources. The figure 6.10 illustrates the principle involved, where all UE communication is done with IEEE 802.21.

The UE initializes the transferring of the QoS configuration, illustrated in the figure 6.11, by sending a notification to the home network PCTS-AS on the subscription setup during initialization (step 1). The notification contains the list of candidate networks discovered with the MIHF, and allows the PCTS-AS to select an appropriate destination network based on the roaming agreements between the home and current networks (step 2). The current PCTS-AS sends the new configuration and the known state of existing multimedia sessions to the PCTS-AS from the target network.

The PCTS-AS in the target network examines the request and makes a policy decision, verifying if the QoS context can be installed in the P-CSCF indicated in the request (step 3). If possible, the authorization request is sent to the new P-CSCF, which will behave as if the service information contained in the request had been derived from an SDP exchange in the target network, contacting the policy and charging control system in IMS to authorize the service information (step 4). The outcome of the authorization request, received back at the P-CSCF, is return on same path to the PCTS-AS in the home network, and finally with a SIP notification to the UE (step 5).

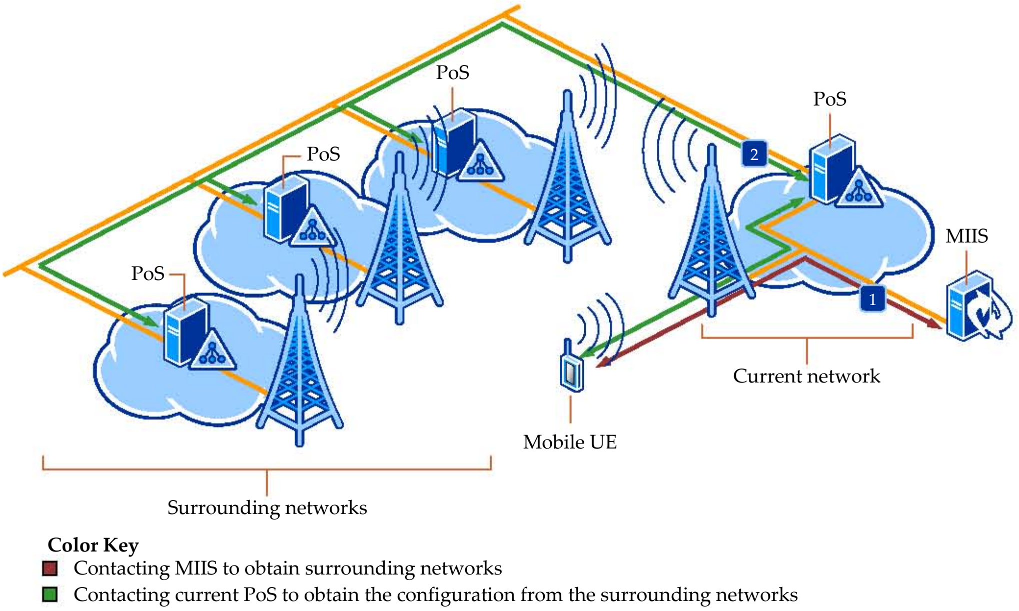

PoS
PoS
PoS
PoS
2
1
MIIS
Current network
Mobile UE
Surrounding networks
Color Key
🟥 Contacting MIIS to obtain surrounding networks
🟩 Contacting current PoS to obtain the configuration from the surrounding networks

Figure 6.10: PCTS context transfer step 1: obtaining the configuration in the target network using the MIHF.

Finally, the current PoS reconfigures the security context in the new network, by transferring the parameters of the IPsec security association to the new PoS, including the ciphering algorithm, the Security Parameter Index, the ports used for the secure communication and the ciphering and integrity keys. This transfer is necessary, such that the UE does not have to re-authenticate itself, and is accomplished by extending the IEEE 802.21 standard. After the reconfiguration completes, the UE may disconnect from the current network and reconnect to the new one, where it completes the previous MIHF request such that the previous PoS releases the resources allocated to the UE, and notifies the home PCTS-AS on the completion of the handover. We note that although the PCTS ensures that all subscribed services are automatically available in the new network, the UE must re-register and update all other sessions.

6.4 Performance Analysis

An estimation of the enhancement benefit of the proactive context transfer is difficult to quantify with a high degree of certainty due to the multitude of network and handover types. In addition, the real-life handover latency is extremely sensitive to the processing times of the signaling messages at the IMS and MIH functions, as well as to the interconnection between different networks. Therefore, in order to estimate the enhancement benefit of the proactive context transfer compared to the previous approaches, we rely on an approximate evaluation method by comparing the individual delay components of each handover type.

In our analysis, we focus exclusively on a UMTS access network, where we measure the handover delay, denoted by T_{ho}, as the duration the video is paused at the corresponding peer (or the home agent in the MIP case). However, our results can be extrapolated to other access technologies, because the access-dependent procedures are similar across all mobility mechanisms. To this end, we identify the main delay components in the table 6.1. In general, the handover delay can be expressed as:

$$T_{ho} = T_{sip} + T_{mip} + T_{attach} + T_{pdpp} + T_{pdps}, \tag{6.1}$$

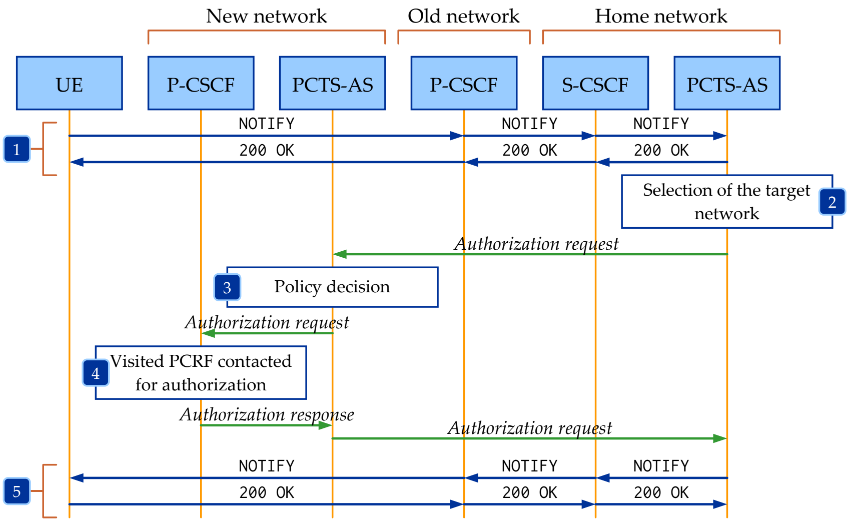


Figure 6.11: PCTS context transfer step 2: network selection.

Delay	Description
T_{attach}	The UE attachment to the UMTS network
T_{pdpp}	The activation of a primary PDP context
T_{pdps}	The activation of a secondary PDP context
T_{sip}	The SIP signaling in the IMS network
$T_{sip,reg}$	The SIP signaling in the IMS network during the registration procedure
$T_{sip,pause}$	The SIP signaling in the IMS network during the stream pause notification
$T_{sip,resume}$	The SIP signaling in the IMS network during the stream resume notification
$T_{sip,update}$	The SIP signaling in the IMS network during the session update
T_{mip}	The Mobile IP message exhange

Table 6.1: Handover delay components of the mobility procedures.

where for each components vary according to the number of signaling messages exchanged and the network entities involved.

6.4.1 Mobility Scenarios

By considering the design specifications of every mobility mechanism, we determine a set of mathematical equations that express the handover delay. This approach helps us estimate the benefit of our proactive context transfer service, without having to consider all real-life circumstances. For the *SIP mobility*, described in the section 6.2.1, we have[2]:

$$\begin{aligned} T_{sip}|_{\text{SIP}} &= T_{sip,pause} + T_{sip,reg} + T_{sip,update} + T_{sip,resume}, \\ T_{mip}|_{\text{SIP}} &= 0. \end{aligned} \tag{6.2}$$

The *optimized SIP mobility* uses the P-CSCF context transfer to reduce the number of signaling steps during the session update after the UE moving to the new network. The new delay, $T'_{sip,update}$, consisting only of a 3-way `INVITE-200 OK-ACK` exchange, as opposed to the standard 8-way session

[2] From the perspective of SIP, the session update is a reinvite procedure.

update, and therefore we have $T'_{sip,update} < T_{sip,update}$. On the other hand, the registration procedure trades the saving of the five SIP messages during update, by introducing an additional message exchange between the old and the new P-CSCF, resulting in a registration delay $T'_{sip,reg} > T_{sip,reg}$. Despite the tradeoff, the overall delay saving is positive, not only due to the difference in the number of signaling messages exchanges, but also because the context transfer involves fewer IMS functions. Hence, we have:

$$\begin{aligned} T_{sip}|\mathrm{SIP}_{opt} &= T_{sip,pause} + T'_{sip,reg} + T'_{sip,update} + T_{sip,resume} < T_{sip}|\mathrm{SIP}, \\ T_{mip}|\mathrm{SIP}_{opt} &= 0. \end{aligned} \tag{6.3}$$

When using *mobile IP*, with the *P-CSCF in the home network*, the UE uses MIP binding messages to request the home agent the buffering or resuming of the video stream, as well as the new IP address in the visited network. In total, four MIP exchanges are synchronous with the UE waiting for the video stream and count toward the handover delay, while the remaining two can be received asynchronously and do not influence the performance. On the IMS side, this handover procedure requires an update of the session, and we obtain:

$$\begin{aligned} T_{sip}|\mathrm{MIP}_{home} &= T_{sip,update}, \\ T_{mip}|\mathrm{MIP}_{home} &= 2T_{mip,rtt}, \end{aligned} \tag{6.4}$$

where $T_{mip,rtt}$ is the round-trip-time of a MIP binding exchange.

As we indicated previously, using MIP with the *P-CSCF in the visited network* requires an extra registration and, therefore, we have:

$$\begin{aligned} T_{sip}|\mathrm{MIP}_{visited} &= T_{sip,reg} + T_{sip,update}, \\ T_{mip}|\mathrm{MIP}_{visited} &= 2T_{mip,rtt}. \end{aligned} \tag{6.5}$$

Although the *proactive context transfer* requires an extensive set of signaling steps before and after roaming, especially in the IEEE 802.21 plane, we remind the reader that these are executed before the interruption of the video stream, or after the download has resumed. Therefore, in estimating the handover delay based on our definition, we use only the steps that are performed while the streaming is stopped. When using *SIP mobility*, the system requires only pausing and resuming notifications, obtaining:

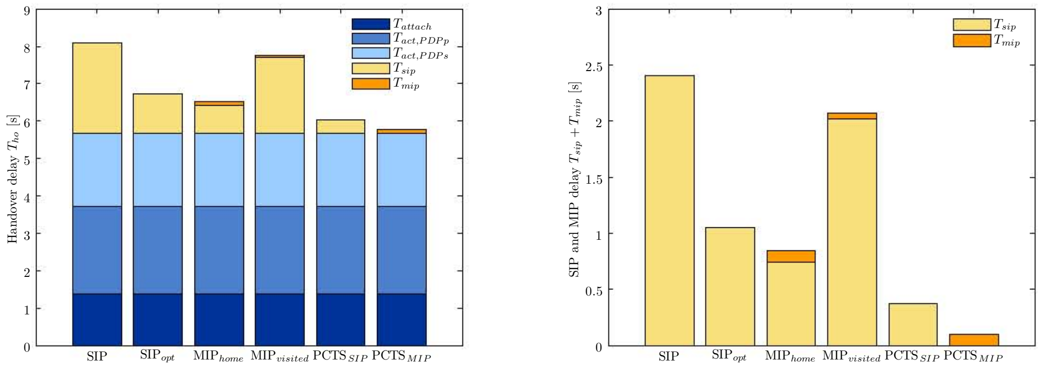
$$\begin{aligned} T_{sip}|\mathrm{PCTS}_{sip} &= T_{sip,pause} + T_{sip,resume}, \\ T_{mip}|\mathrm{PCTS}_{sip} &= 0. \end{aligned} \tag{6.6}$$

The *mobile IP* alternative uses MIP messages for the same purpose, and we have:

$$\begin{aligned} T_{sip}|\mathrm{PCTS}_{sip} &= 0, \\ T_{mip}|\mathrm{PCTS}_{sip} &= T_{mip,rtt}. \end{aligned} \tag{6.7}$$

6.4.2 Handover Delay Comparison

By using measurement data on the delay of SIP and MIP exchanges in an UMTS network, we can attach approximative numeric values to our analytic estimation. To this end, Pesch et al. offer



(a) The handover delay for a UE roaming between UMTS access networks.

(b) The SIP and MIP component of the handover delay.

Figure 6.12: Analysis of the handover delay.

an extensive set of SIP delay measurements (Pesch et al., 2005). We extrapolate their data to our situations, considering the number of network entities interacting and the number of signaling messages sent/received.

For the delay of specific SIP messages in the UMTS access network, we developed an implementation of the IPTV SIP stack using the Java programming language and the JAIN-SIP API[3]. Our test-bed measured the round-trip-times at intervals of 15 minutes during a period of one month. By relying on this experimental data, in the figure 6.12(a), we show the handover delay for each scenario. The network access delay, which includes the GPRS attachment and the establishment of PDP contexts, is common to all mechanisms. For this reason, the figure 6.12(b) emphasizes only the SIP and MIP signaling delay, which is specific to the methods we presented. With this last result, it becomes obvious that the PCTS approach offers a significant reduction in the handover delay, by effectively executing most of the handover signaling before the UE detaches the old network.

6.5 Conclusions

In this chapter, we surveyed the mobility support for the P2PTV IMS service. To this end, we studied existing approaches to mobility, based on both SIP and MIP signaling. Since the most expensive operation of the handover procedure is the registration and the sessions update in the new network, we conclude that a proactive approach provides a significantly superior performance. The main idea is to use the existing IEEE 802.21 Media Independent Handover technology, to detect beforehand the UE configuration in the new network. For this purpose, we propose a new IMS service, called a *proactive context transfer service*, which facilitates the discovery of the target network settings, and the transfer of the UE's current state to the IMS entities of the same network.

The enhancement offered by the PCTS can be tailored to the actual requirements of each multimedia service. Therefore, services for which continuous connectivity is essential for the user experience, such as audio/video streaming, may take advantage of the PCTS functionality, while other services could continue using the classic signaling. In order to work, our proposal requires some new changes to the IEEE 802.21 and IMS specifications, which facilitate the transfer of state information between functional entities.

The benefits of the PCTS are a significant reduction in the handover delay, measured as the interruption of the P2PTV download. Our experimental evaluation shows that the SIP/MIP

[3] JAIN SIP Developer Tools, https://jain-sip.dev.java.net/ .

delay component of the handover (that is excluding any access network connectivity procedures) decrease from over 2 seconds in the case of the standards down to several hundreds of milliseconds. This reduction is important because it reduces the level of buffer usage at the uploading peers, and, most importantly, reduces the initial buffering delay during playback.

Part

III

The User Activity in a Large-Scale IPTV System

Chapter 7

Workload Model

7.1 Overview

A successful design of the P2PTV application server session and peer coordination algorithms must consider carefully the activity of the TV subscribers. In particular, to determine the optimal number of upload and download sessions and their distribution across the UE peers, finding the nature of the TV channel switching has a paramount importance. If the application server can predict, at least in the short term, the channel switching process, it can pro-actively allocate the necessary SIP sessions such that new channel users are accommodated with a minimum delay.

Unfortunately, measuring and modeling a real-life large-scale IPTV system is far from a simple task. The most important obstacle to a thorough understanding of these systems is the lack of actual data showing how users watch TV, due to the closed architecture of the existing deployments. For these reasons, many IPTV measurement studies focused on the Internet-based services such as PPLive, PPStream, SopCast, TVAnts (Hei et al., 2007; Silverston & Fourmaux, 2007; Ali et al., 2006; Xie et al., 2007; Vu et al., 2007). These services provide an experience completely different from the classic television, in that users access the TV channels using a browsing/viewing software installed on their computers. In addition, the technological constraints such limited bandwidth, peer-to-peer streaming, large channel connection delay etc., preclude users to switch or browse TV channels at the same pace as they would with a normal TV set.

While these studies generate an excellent insight into the behavior of Internet IPTV users, we find them inadequate for our work, which targets a commercial walled-garden environment. For this reason, we focus on the narrow set of measurements from telco-deployed IPTV. Cha et al. (Cha, Rodriguez, Crowcroft, et al., 2008) present the first complete study of the user activity in a large IPTV provider, using anonymized data collected from the Telefonica IPTV service, Movistar Imagenio. Qiu et al. (Qiu, Ge, Lee, Wang, Xu, & Zhao, 2009; Qiu, Ge, Lee, Wang, Zhao, & Xu, 2009) published a similar result using traces collected from AT&T U-verse service, and furthermore, they propose a workload generation model for an IPTV system called *Simulwatch*.

Their model offers a good start for generating a synthetic workload. However, in order to keep

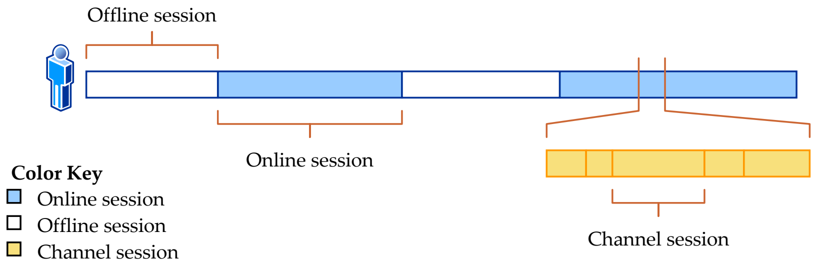


Figure 7.1: Modeling the user activity.

its complexity low, it suffers several shortcomings. First, the authors limit the number of system properties they model, excluding, for example, the weekly variations of active users. Second, their published parameters often yield less than excellent matches between the model and the actual data. Finally, their model has a few flaws.

In our work, we rely on their published data to enhance the Simulwatch model, such that the synthetic workload better matches the data trace. Specifically, because our objective is to use this model to drive the performance evaluation of our P2PTV service, we do not intend to limit the number of parameters that characterize the system at the expense of a less than perfect match. Like Simulwatch, our model focuses on the following key workload aspects: the *user activity* and the *channel popularity*.

- The *user activity* includes two characteristics: (i) the *rate* at which users change the state of their viewing activity called *session*, i.e. turn on and off their user equipment, and change between channels; and (2) the *length* of each on, off or channel session. In addition, these characteristics will determine the number of online UEs at a given time.
- The *channel popularity* indicates the channel a user is most likely to watch. The popularity metric has two dimensions: temporal and user space. The temporal dimension refers to the long-term mean and short-term variations in the popularity of the channel. The user dimension models the user preference for certain TV channels.

7.2 User Activity

The user activity consists in three types of sessions:

- *Online sessions* when the user's user equipment (UE) is turned on, and it can actively participate in forwarding the TV channel to other users.
- *Offline sessions* when the user's UE is turned off.
- *Channel sessions* when the user is watching a particular TV channel. Channel sessions overlap with the UE online sessions.

The figure 7.1 illustrates the types of sessions for a user equipment. The parameters from the available data, which characterize the dynamics of the user activity, are the following:

- The *session length* of each session type, denoted by d_{on}, d_{off}, and d_{ch}, respectively.
- The *session rate* of each session type, denoted by x_{on}, x_{off} and x_{ch}, respectively.
- The *ratio of online and offline UEs*, normalized to the total number of active users from the system: r_{on} and r_{off}.

We emphasize these parameters in the figure 7.2, where they are generated by the superposition of multiple viewers watching TV at the same time.

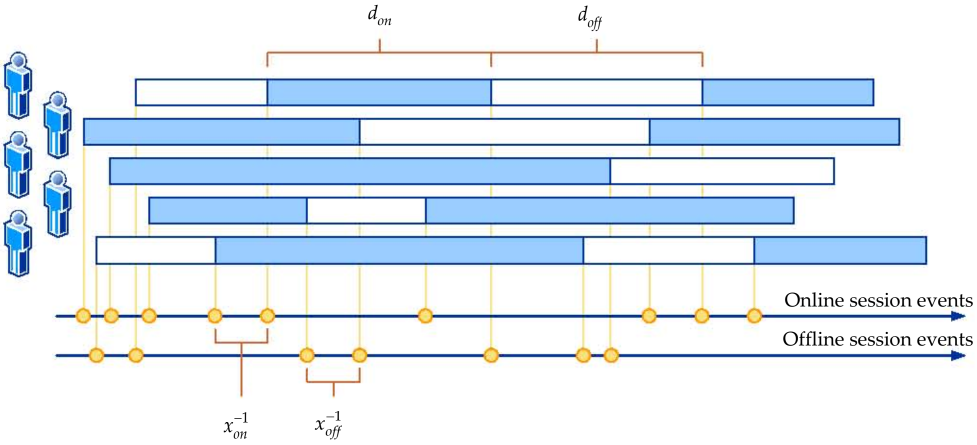


Figure 7.2: Parameters that characterize the user activity.

7.2.1 Session Length

The sessions from the user activity represent a *renewal process*, characterized by a probability distribution for each session type. Qiu et al. propose modeling the distribution of this empirical data using a *mixture-exponential* or *hyper-exponential* probability distribution. Unfortunately, their parameters offers only a rough match to the data trace, which is emphasized in the model evaluation. For this reason, in this section, we enhance the fitting between the data and the synthetic model.

The hyper-exponential can approximate a large variety of long-tail probability distributions (Jewell, 1982). Among the few conditions for successful fitting is that the probability distribution it approximates must have a strictly monotonic probability density function (PDF). Therefore, distributions such as normal cannot be approximated with a hyper-exponential. Since an analysis of the available data indicates that its PDF is monotonic, its probability distribution can be successfully approximated with a hyper-exponential, with the CDF of the form:

$$F(d) = 1 - \sum_{i=1}^{n} p_i \mathrm{e}^{-\lambda_i d}, \tag{7.1}$$

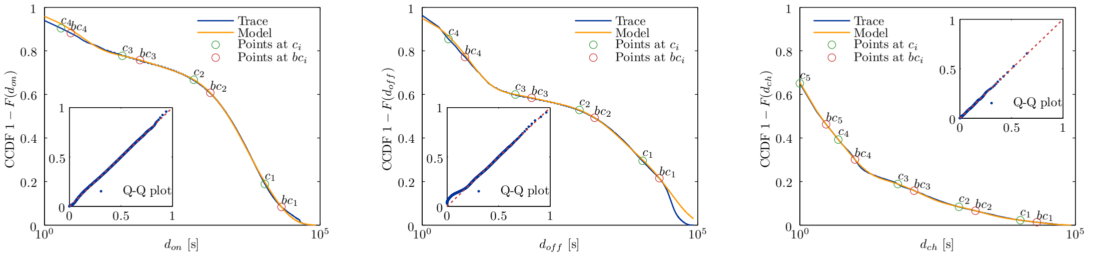
where $x \in [0, \infty)$, $\lambda_i > 0$ and:

$$\sum_{i=1}^{n} p_i = 1. \tag{7.2}$$

In order to determine the parameters of each hyper-exponential distribution (for d_{on}, d_{off}, and d_{ch}), we use the fitting algorithm described by Feldmann et al. (Feldmann & Whitt, 1998). The method consists in fitting each exponential from the mixture to exponentially spaced intervals $[c_i, b \cdot c_i]$, where b is a factor representing the interval size, typically 2.

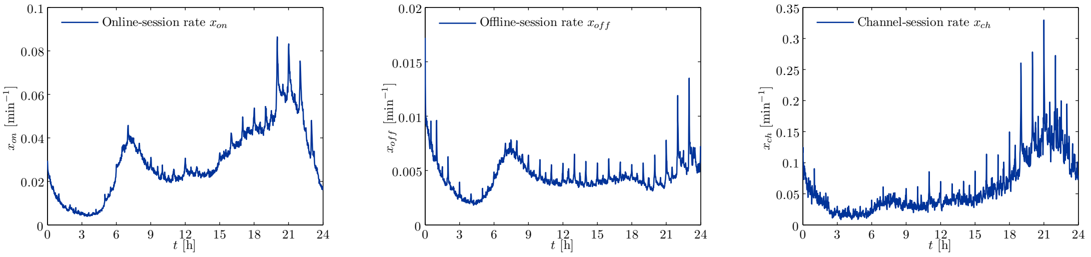
We apply this procedure for the data from all session types. The figures 7.3(a)–7.3(c) illustrate the hyper-exponential fitting for each session length. We selected $n = 4$ for d_{on}, d_{off}, $n = 5$ for d_{ch}, and we emphasized the interval points used for the fitting (c_i and bc_i). By comparing the data and the model distributions in a Q-Q plot, we observe that we obtain a reasonable god fit in every case. For the channel session length, we have used a larger number of parameters to characterize the model, due to the large fraction of sessions of less than 1 second, and the lack of data for these sessions. Therefore, all sessions in the 0–10 second range are approximated by two exponentials in the mixture.

The table 7.1 illustrates the parameters of the model. The Kolmogorov-Smirnov (K-S) statistic



(a) On-session length d_{on}. (b) Off-session length d_{off}. (c) Channel-session length d_{ch}.

Figure 7.3: Fitting the session length data to a hyper-exponential distribution.



(a) Online session rate x_{on}. (b) Offline session rate x_{off}. (c) Channel session rate x_{ch}.

Figure 7.4: User session rate during a typical day.

confirms the goodness-of-fit in every case.

Parameter	d_{on}	d_{off}	d_{ch}	Parameter	d_{on}	d_{off}	d_{ch}
p_1	0.4264	0.4030	0.03876	$\lambda_1 \cdot 10^5$	8.1219	3.0985	5.3894
p_2	0.3110	0.1745	0.07252	$\lambda_2 \cdot 10^4$	3.6078	3.691	5.6563
p_3	0.0647	0.0460	0.1240	$\lambda_3 \cdot 10^2$	1.518	1.203	0.7326
p_4	0.1980	0.3763	0.3499	λ_4	0.2301	0.1414	0.15411
p_5	-	-	0.4148	λ_5	-	-	1.3115

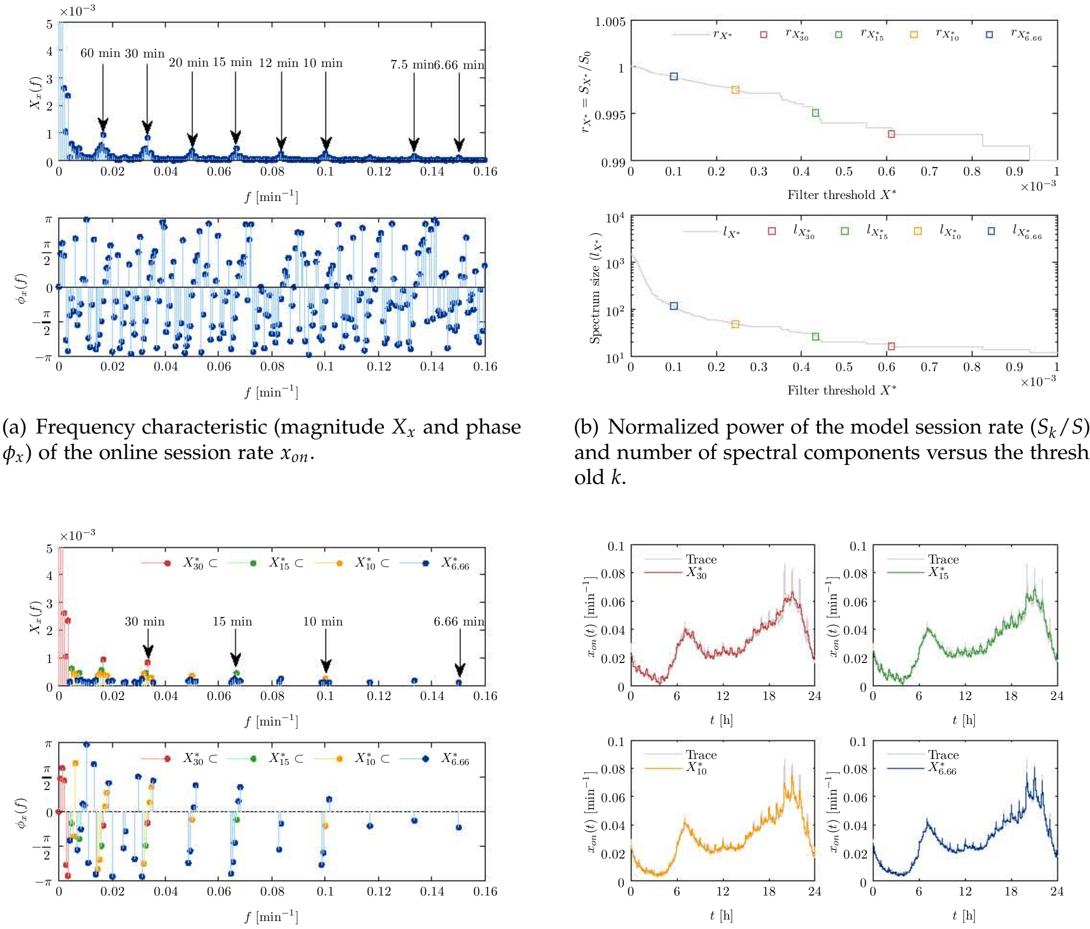
Table 7.1: Session length model: the parameters of the hyper-exponential distribution.

7.2.2 Session Rate

In an IPTV system the user activity sessions are not independent random variables. Instead, they are all conditioned by an external process: the TV program schedule (Qiu, Ge, Lee, Wang, Xu, & Zhao, 2009) and diurnal user activity. Therefore, the superposition of the activity sessions for all users in the system is not an exclusively stochastic process over the long term.

7.2.2.1 Modeling the Daily Pattern

The figures 7.4(a)–7.4(c) illustrate the rate at which all users in the system turn their TV equipments on and off, and change between channels during a typical day. The data is sampled at one minute intervals. The on/off session rate represents the number of UEs turned on/off during a sample interval, normalized to the number of off/on UEs at the beginning of the same interval. The channel session rate is the number of channel changes during a sampling interval, normalized to the number of online UEs during the same interval. The figures show that while the session rate



(a) Frequency characteristic (magnitude X_x and phase ϕ_x) of the online session rate x_{on}.

(b) Normalized power of the model session rate (S_k/S), and number of spectral components versus the threshold k.

(c) The left half of the session-rate spectrum approximated by the model.

(d) Comparing the reconstructed session-rate for models with different threshold values and number of parameters.

Figure 7.5: Modeling the daily session rate with a frequency analysis.

is seemingly complex, it also exhibits an intuitive pattern based on increased user activity in the morning and in the evening.

In order to model the session rate with such a high degree a complexity in the time-domain, Qiu et al. (Qiu, Ge, Lee, Wang, Xu, & Zhao, 2009) propose a different approach based on a frequency analysis. However, while their pioneering idea has merit, it does not capture the complexity of the data well enough. For this, reason, we extend their model with a similar but more thorough approach. In addition, we take our model one step further by integrating the weekly activity pattern.

Since the method we present is the same for the rates of all three session types, denoted by x_{on}, x_{off}, x_{ch}, we shall illustrate the technique only for one of them. By relying on the Fourier analysis, our objective is to find a small-enough set of spectral components that can accurately characterize the time-domain function. The figures 7.5(a)–7.5(d) show that the session rate has a small number of dominant spectral components, centered around certain time intervals at 60 minutes, 30 minutes, 15 minutes etc, corresponding to the TV program. In addition, several low-frequency high-amplitude components describe the diurnal pattern.

Unlike the Qiu model, we do not attempt to approximate the session rate spectrum with a continuous function. First, we understand that any approximation of important spectral components has a serious effect in time-domain, and, second, the phase information (which the authors of (Qiu, Ge, Lee, Wang, Xu, & Zhao, 2009) do not model at all) cannot be easily approximated by any know function. Instead, our model resorts to identifying the dominant frequency components.

Toward this end, we apply a threshold-based approach for the spectrum magnitude, where the threshold value X^* is selected based on (i) the selected components contain a reasonable fraction of the signal power, (ii) the model encompasses the desired number of dominant spectral components, and (iii) the number of spectral components and hence model parameters is small. In order to determine a good threshold, in the figure 7.5(b), we show the power ratio of the modeled signal, denoted by r_{X^*} for a variable threshold X^*. We have:

$$r_{X^*} = \frac{S_{X^*}}{S_0}, \tag{7.3}$$

where S_{X^*} is the power of the modeled online session rate, containing only frequency components of magnitude $X \geq X^*$, and S_0 is the power of the whole function.

For a threshold that satisfies the last two conditions, we select a magnitude threshold that contain a relevant set of frequency components. From the set of dominant frequencies identified in the figure 7.5(a), we choose a set of threshold values that progressively include the components at 30, 15, 10 and 6.66 minutes. We denote these thresholds by X^*_{30}, X^*_{15}, X^*_{10}, and $X^*_{6.66}$. For any smaller threshold, our model obtains an increasingly better approximation of the real trace, at the expense of a larger number of model parameters.

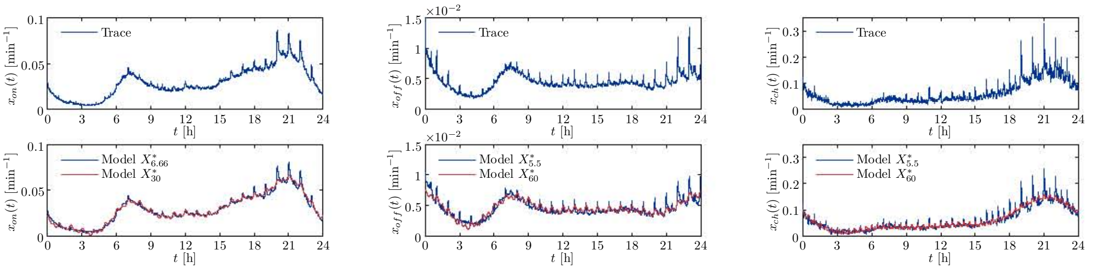
Threshold X^*	Spectrum size l_{X^*}	Power ratio r_{X^*}	Model parameters (real + complex)
X^*_{30}	14+1	99.28 %	9+7
X^*_{15}	24+1	99.51 %	14+12
X^*_{10}	46+1	99.75 %	25+23
$X^*_{6.66}$	114+1	99.89 %	59+57

Table 7.2: Online session spectrum characteristics for the selected threshold values.

The table 7.2 contains the spectrum size, power ratio and model parameters that we obtained for each threshold. The figure 7.5(c) illustrates the number of spectrum samples that are *incrementally* included in the model with an increasingly smaller threshold, X^*, i.e. samples for a large X^* are also part of the spectrum of a small X^*. While a smaller threshold adds more accuracy to the model, it also increases the number of model parameters. We note that each spectral component, magnitude and phase, represents the same complex number, and the spectrum is symmetric in magnitude and anti-symmetric in phase, except for the continuous component at frequency zero. Hence, for a number of l_{X^*} frequencies in the spectrum, including the continuous component, the model will have $(l_{X^*} - 1)/2$ complex parameters describing each spectral component at $f > 0$, and $(l_{X^*} + 3)/2$ real parameters describing the frequency value and the magnitude of the continuous component.

The figure 7.5(d) summarizes the online session rate, x_{on}, generated from our frequency model by using different threshold values and number of parameters. We can observe that a model that uses a lower threshold X^* and a larger set of parameters approximates better the original data. Therefore, choosing a relevant threshold depends on obtaining a desired accuracy, especially for highly transient events.

In the figures 7.6(a)–7.6(c), we illustrate and compare our frequency model for each type of session with the original trace data. For the model, we selected two thresholds, a high and a low one. The figure demonstrates the trade-off between the number of parameters that characterize the model and its accuracy.



(a) Online session rate x_{on} model. (b) Offline session rate x_{off} model. (c) Channel session rate x_{ch} model.

Figure 7.6: Approximation of the daily session rate with our frequency model.

7.2.2.2 Determining the Number of Online/Offline UEs

We do not model the number of online/offline UEs directly and instead, they are indirectly modeled through the session rate. For the sake of simplicity, we shall normalize all values such that the sum of all UEs in the system is 1. Hence, we can refer to the number of online/offline UEs as the *online/offline UE ratio*, denoted by r_{on} and r_{off}, respectively. This approach has the advantage of extending easily to any arbitrary number of UEs. Hence:

$$r_{on} + r_{off} = 1. \tag{7.4}$$

According to the available data from Qiu et al., the online/offline session rate measures the number of UEs that are turned on/off during a 1 minute interval, normalized to the number of UEs in an offline/online state at the beginning of the same interval.

Corollary 1. *Let us assume an arbitrary Δt interval ending at time t. If the online/offline UE ratios at the beginning of the interval are r_{on} and r_{off}, respectively, and the online/offline session rates during this interval are $r_{on}(t)$ and $r_{off}(t)$, respectively, then the fraction of UEs that are online/offline at the end of the interval are:*

$$\begin{aligned} r_{on}(t) &= r_{on}(t-\Delta t) + x_{on}(t) r_{off}(t-\Delta t)\Delta t - x_{off}(t) r_{on}(t-\Delta t)\Delta t, \\ r_{off}(t) &= r_{off}(t-\Delta t) + x_{off}(t) r_{on}(t-\Delta t)\Delta t - x_{on}(t) r_{off}(t-\Delta t)\Delta t. \end{aligned} \tag{7.5}$$

Proof. From the aforementioned definition of the on/off session rate, it follows that the fraction of *off* UEs that are turned on is $x_{on} r_{off} \Delta t$. Similarly, the fraction of *on* UEs that are turned off is $x_{off} r_{on} \Delta t$.

Hence, the solution follows. □

Theorem 1. *Let an initial state of the system at time $t = 0$ with a $r_{on}(0)$ and $r_{off}(0)$ online/offline UE ratio, respectively. Given a sufficiently large $t > 0$, the online/offline UE ratio at time t, $r_{on}(t)$ and $r_{off}(t)$, does not depend on $r_{on}(0)$ and $r_{off}(0)$, and their values are determined solely by the online/offline session rate.*

Proof. We shall prove the theorem for $r_{on}(t)$. The proof for $r_{off}(t)$ is complementary.

From equations 7.5 and 7.4 it follows:

$$\begin{aligned} r_{on}(t) &= r_{on}(t-\Delta t) + x_{on}(t)\left(1 - r_{on}(t-\Delta t)\right)\Delta t - x_{off}(t) r_{on}(t-\Delta t)\Delta t, \\ &= r_{on}(t-\Delta t)\left(1 - x_{on}(t)\Delta t - x_{off}(t)\Delta t\right) + x_{on}(t)\Delta t. \end{aligned} \tag{7.6}$$

Let us denote for simplicity the following coefficients:

$$\begin{aligned} A(t) &= 1 - x_{on}(t)\Delta t - x_{off}(t)\Delta t, \\ B(t) &= x_{on}(t)\Delta t. \end{aligned} \tag{7.7}$$

Therefore, the fraction of online UEs at time t is determined by the recursive equation:

$$r_{on}(t) = A(t) \cdot r_{on}(t - \Delta t) + B(t). \tag{7.8}$$

To solve this equation, there exists a Δt for which we can write:

$$\begin{aligned} r_{on}(t) &= A(t) \cdot r_{on}(t - \Delta t) + B(t), \\ r_{on}(t - \Delta t) &= A(t - \Delta t) \cdot r_{on}(t - 2\Delta t) + B(t - \Delta t), \\ r_{on}(t - 2\Delta t) &= A(t - 2\Delta t) \cdot r_{on}(t - 3\Delta t) + B(t - 2\Delta t), \\ &\vdots \\ r_{on}(\Delta t) &= A(\Delta t) \cdot r_{on}(0) + B(\Delta t). \end{aligned} \tag{7.9}$$

By substituting, we obtain:

$$r_{on}(t) = r_{on}(0) \cdot \prod_{\substack{u=\Delta t \\ u \leftarrow u+\Delta t}}^{t} A(u) + \sum_{\substack{u=\Delta t \\ u \leftarrow u+\Delta t}}^{t} B(u) \prod_{\substack{v=u+\Delta t \\ v \leftarrow v+\Delta t}}^{t} A(v). \tag{7.10}$$

From the definition of the online/offline session rate, and unless the system is inactive[1], for a t and Δt, we have:

$$0 < x_{on}(t)\Delta t + x_{off}(t)\Delta t < 2, \tag{7.11}$$

because there cannot exist a number of UEs changing state from online-to-offline and offline-to-online, during an arbitrary Δt, larger than the total number of UEs in the system[2].

Given this condition, it results that $-1 < A(t) < 1$, and when $t \gg 0$ the first product in equation 7.10 tends to zero. Therefore, when $t \gg 0$, we obtain that:

$$r_{on}(t) \approxeq \sum_{\substack{u=\Delta t \\ u \leftarrow u+\Delta t}}^{t} B(u) \prod_{\substack{v=u+\Delta t \\ v \leftarrow v+\Delta t}}^{t} A(v), \tag{7.12}$$

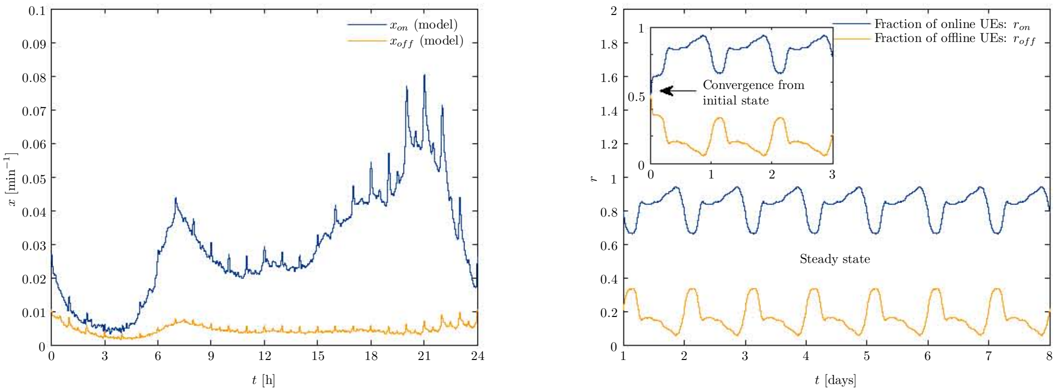
which does not depend on the initial state of the system.

□

The figures 7.7(a) and 7.7(b) give an exceptional insight into the findings from the previous theorem. We simulated the evolution of the IPTV system using the frequency model of the session rate, illustrated in the figure 7.7(a), and the recursive equation of the of online/offline UE ratio. At the initial moment, the UEs were divided in half online and half offline, $r_{on}(0) = r_{off}(0) = 0.5$. The result shows that, except for an initial transition period during which the first product from equation 7.10 tends to zero, the UE ratios r_{on} and r_{off} depend only on x_{on} and x_{off}. For the same reason, both variables have a strong diurnal pattern corresponding to the session rate.

[1] That is $x_{on}(t) = x_{off}(t) = 0$ for any t.

[2] A UE is considered to have changed state if at the end of Δt interval has a different state than at the beginning. It does not matter if the same UE changes state multiple times during Δt. The condition simply states that at the end of Δt we cannot have more UEs in both on and off states, than the number we had at the beginning. At the extreme, if all UEs change state, then $x_{on}(t)\Delta t = x_{off}(t)\Delta t = 1$. However, this extreme cannot happen for any Δt, as there exists always a smaller Δt for which at least one UE does not change the state.



(a) Online/offline session rate $x_{on}(t)$, $x_{off}(t)$ obtained with our frequency model.

(b) Evolution of the online/offline UE ratios obtained with the modeled session rate.

Figure 7.7: Online/offline UE ratios: $r_{on}(t)$, $r_{off}(t)$.

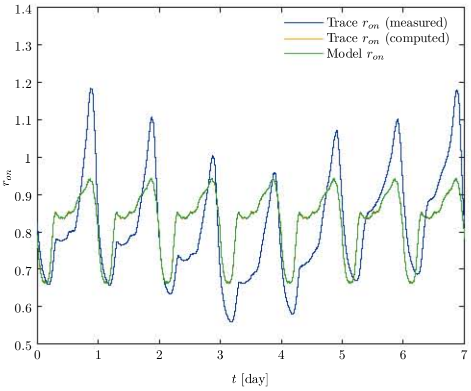


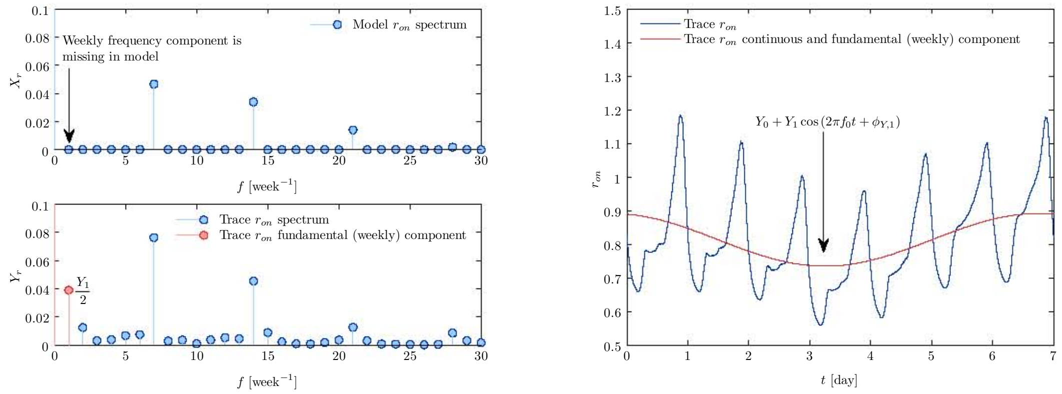
Figure 7.8: Comparing the online UE ratio between the real trace and the model.

7.2.2.3 Modeling the Weekly Pattern

In the figure 7.8, we compare the model with the trace data over a period of one week, re-normalized to the mean obtained from the model. While our model generates an online/offline UE ratios that is similar to the real trace, two main differences become obvious. First, the TV usage peak in the evening exceeds slightly our predicted value. Second, there exists a clear weekly pattern in the number of online UEs, corresponding to an increases TV usage in the weekend. The trace r_{on} has been obtained in two ways: measured directly and computed from the session rate, similar to our model. The first day of the week, day zero, is Sunday.

The figure illustrates that our model approximates extremely well the computed user ratio, however there exists a larger difference to the one measured directly. The reason for these differences lie in a subtle understanding of the limitations of the measured data. First, the published data in an approximation of the real-life TV usage and has limited time resolution. For instance, the time resolution of the session rate is one minute, and therefore valuable information on the dynamics of the system, at smaller time scales is lost. Second, the parameters of the session rate model were determined by examining trace data from a typical day. However, as typical as that day may be, the measured data does not capture the subtle variation across large time scales, such as weeks.

Since the first issue can only be solved by more accurate measurements of a deployed IPTV system, in this section, we modify our model to address the weekly pattern of the number



(a) Comparing the frequency characteristic between model X and trace Y.

(b) Evolution of the online/offline UE ratio, obtained with the modeled session rate.

Figure 7.9: The frequency analysis of the online UE ratio data.

of online/offline UEs. Since the online/offline UE ratios are modeled indirectly through the online/offline session rate, we must modify the session rate model to accommodate the weekly variations. The method we propose is the following.

- We preform a frequency analysis on the data trace to identify the weekly variations in the number of online UEs.
- We propose a modulating function $M(t)$ for the online UE ratio $r_{on}(t)$ that approximates the measured data.
- We compute the new online/offline session rates, denoted by $r^*_{on}(t)$ and $r^*_{off}(t)$.

By applying a discrete Fourier transform on the trace data, we determine the main frequency components in the number of online UEs. We denote the frequency characteristic of the model r_{on} by X, and of the trace r_{on} by Y. In the figure 7.9(a), comparing the two spectra, we immediately observe that the trace data has an additional frequency component with a weekly period. Let us denote this weekly frequency by f_0, and the weekly component magnitude and phase by Y_1 and $\phi_{Y,1}$, respectively. In addition, we denote by Y_0 the continuous component of the trace r_{on}, and by Y_i the magnitude of the component at frequency $i \cdot f_0$. The figure 7.9(b) shows the same components in time domain.

Based on the frequency analysis, we can write the model and trace r_{on} in terms of their dominant frequency components, as follows.

$$\begin{aligned}
r_{on}(t)|_{\text{model}} &= X_0 \\
&+ X_7 \cos(2\pi 7 f_0 t + \phi_{X,7}) \\
&+ X_{14} \cos(2\pi 14 f_0 t + \phi_{X,14}) \\
&+ X_{21} \cos(2\pi 21 f_0 t + \phi_{X,21}) + \cdots, \\
r_{on}(t)|_{\text{trace}} &\approxeq Y_0 \\
&+ Y_1 \cos(2\pi f_0 t + \phi_{Y,1}) \\
&+ Y_7 \cos(2\pi 7 f_0 t + \phi_{Y,7}) \\
&+ Y_{14} \cos(2\pi 14 f_0 t + \phi_{Y,14}) \\
&+ Y_{21} \cos(2\pi 21 f_0 t + \phi_{Y,21}) + \cdots.
\end{aligned} \tag{7.13}$$

In order to obtain the trace online UE ratio from the modeled one, we propose a modulating function $M(t)$, such that:

$$r_{on}(t)|_{\text{trace}} = M(t) \cdot r_{on}(t)|_{\text{model}}, \tag{7.14}$$

where the modulating function has the form:

$$M(t) = M_0 + M_1 \cos(2\pi f_0 t + \phi_M). \tag{7.15}$$

The rationale for choosing a modulating function, rather that simply adding the fundamental component to the model, is that a product facilitates the transformation of the online/offline session rate. The result we obtain is an approximative one, due to (i) the difficulty of matching the amplitude and phase of all dominant spectral components, and (ii) the appearance of spectral components around the dominant frequencies. However, since it is essential to model the number of UEs through the session rate, the modulating function has only three parameters, and because the error is small, we feel that this approach is a good compromise[3].

By modulating the model $r_{on}(t)$ with $M(t)$, we obtain:

$$\begin{aligned} r_{on}(t)|_{\text{trace}} &= M(t) \cdot r_{on}(t)|_{\text{model}} \\ &= X_0 M_0 \\ &+ X_0 M_1 \cos(2\pi f_0 t + \phi_{X,1}) \\ &+ X_7 M_0 \cos(2\pi 7 f_0 t + \phi_{X,7}) \\ &+ \frac{1}{2} X_7 M_1 \cos(2\pi 6 f_0 t + \phi_{X,7} - \phi_M) + \frac{1}{2} X_7 M_1 \cos(2\pi 8 f_0 t + \phi_{X,7} + \phi_M) \\ &+ X_{14} M_0 \cos(2\pi 14 f_0 t + \phi_{X,14}) \\ &+ \frac{1}{2} X_{14} M_1 \cos(2\pi 13 f_0 t + \phi_{X,14} - \phi_M) + \frac{1}{2} X_{14} M_1 \cos(2\pi 15 f_0 t + \phi_{X,14} + \phi_M) + \cdots . \end{aligned} \tag{7.16}$$

By identifying the magnitude and phase of the most important frequencies, we find the parameters of the modulating function:

$$\begin{aligned} M_0 &= \frac{Y_0}{X_0}, \\ M_1 &= \frac{Y_1}{X_0}, \\ \phi_M &= \phi_{Y,1}. \end{aligned} \tag{7.17}$$

However, because of the approximations in the model, we must be careful that the modulated session rates do not exceed the interval $[0, 1]$. If these boundaries are exceeded, we have to modify the modulating function accordingly.

Theorem 2. *Let $r_{on}(t)$ and $r_{off}(t)$ be the online/offline UE ratios, obtained from the online/offline session rates, x_{on} and x_{off}. Given the continuous modulating function $M(t)$, during the steady state of the system we obtain a new online UE ratio $r^*_{on}(t) = M(t) r_{on}(t)$ and an offline UE ratio $r^*_{off}(t) = 1 - r^*_{on}(t)$ when the new online/offline session rates are:*

[3] It is possible to propose a more complex modulating function with spectral components at multiple frequencies.

$$\begin{aligned} x^*_{on}(t) &= M(t)x_{on}(t), \\ x^*_{off}(t) &= (1 - M(t))x_{on}(t) + x_{off}(t). \end{aligned} \tag{7.18}$$

Proof. From the theorem 1, the online UE ratio is:

$$r_{on}(t) = r_{on}(0) \prod_{\substack{u=\Delta t \\ u \leftarrow u+\Delta t}}^{t} A(u) + \sum_{\substack{u=\Delta t \\ u \leftarrow u+\Delta t}}^{t} B(u) \prod_{\substack{v=u+\Delta t \\ v \leftarrow v+\Delta t}}^{t} A(v), \tag{7.19}$$

where:

$$\begin{aligned} A(t) &= 1 - x_{on}(t)\Delta t - x_{off}(t)\Delta t, \\ B(t) &= x_{on}(t)\Delta t. \end{aligned} \tag{7.20}$$

Similarly, given the new session rates $x^*_{on}(t)$, $x^*_{off}(t)$, the modulated online UE is:

$$r^*_{on}(t) = r^*_{on}(0) \prod_{\substack{u=\Delta t \\ u \leftarrow u+\Delta t}}^{t} A^*(u) + \sum_{\substack{u=\Delta t \\ u \leftarrow u+\Delta t}}^{t} B^*(u) \prod_{\substack{v=u+\Delta t \\ v \leftarrow v+\Delta t}}^{t} A^*(v), \tag{7.21}$$

where:

$$\begin{aligned} A^*(t) &= 1 - x^*_{on}(t)\Delta t - x^*_{off}(t)\Delta t, \\ B^*(t) &= x^*_{on}(t)\Delta t. \end{aligned} \tag{7.22}$$

We have:

$$r^*_{on}(t) = M(t)r_{on}(t) = r_{on}(0)M(t) \prod_{\substack{u=\Delta t \\ u \leftarrow u+\Delta t}}^{t} A(u) + M(t) \sum_{\substack{u=\Delta t \\ u \leftarrow u+\Delta t}}^{t} B(u) \prod_{\substack{v=u+\Delta t \\ v \leftarrow v+\Delta t}}^{t} A(v), \tag{7.23}$$

where we identify:

$$\begin{aligned} \prod_{\substack{u=\Delta t \\ u \leftarrow u+\Delta t}}^{t} A^*(u) &= M(t) \prod_{\substack{u=\Delta t \\ u \leftarrow u+\Delta t}}^{t} A(u), \\ \sum_{\substack{u=\Delta t \\ u \leftarrow u+\Delta t}}^{t} B^*(u) \prod_{\substack{v=u+\Delta t \\ v \leftarrow v+\Delta t}}^{t} A^*(v) &= M(t) \sum_{\substack{u=\Delta t \\ u \leftarrow u+\Delta t}}^{t} B(u) \prod_{\substack{v=u+\Delta t \\ v \leftarrow v+\Delta t}}^{t} A(v) \end{aligned} \tag{7.24}$$

From the fist equality, we obtain that:

$$\begin{aligned} & A^*(\Delta t) = M(\Delta t)A(\Delta t), \\ & A^*(\Delta t)A^*(2\Delta t) = M(2\Delta t)A(\Delta t)A(2\Delta t) && \Rightarrow \quad A^*(2\Delta t) = \frac{M(2\Delta t)}{M(\Delta t)}A(2\Delta t), \\ & A^*(\Delta t)A^*(2\Delta t)A^*(3\Delta t) = M(3\Delta t)A(\Delta t)A(2\Delta t)A(3\Delta t) && \Rightarrow \quad A^*(3\Delta t) = \frac{M(3\Delta t)}{M(2\Delta t)}A(3\Delta t), \\ & && \quad \vdots \end{aligned} \tag{7.25}$$

and in general:

$$A^*(t) = \frac{M(t)}{M(t-\Delta t)} A(t). \tag{7.26}$$

By replacing the last equality in the second part of equation 7.24, we obtain:

$$\begin{aligned}
\sum_{\substack{u=\Delta t\\ u\leftarrow u+\Delta t}}^{t} B^*(u) \prod_{\substack{v=u+\Delta t\\ v\leftarrow v+\Delta t}}^{t} A^*(v) &= \sum_{\substack{u=\Delta t\\ u\leftarrow u+\Delta t}}^{t} B^*(u) \prod_{\substack{v=u+\Delta t\\ v\leftarrow v+\Delta t}}^{t} \frac{M(v)}{M(v-\Delta t)} A(v) \\
&= \sum_{\substack{u=\Delta t\\ u\leftarrow u+\Delta t}}^{t} B^*(u) \frac{M(t)}{M(u)} \prod_{\substack{v=u+\Delta t\\ v\leftarrow v+\Delta t}}^{t} A(v) \\
&= M(t) \sum_{\substack{u=\Delta t\\ u\leftarrow u+\Delta t}}^{t} \frac{B^*(u)}{M(u)} \prod_{\substack{v=u+\Delta t\\ v\leftarrow v+\Delta t}}^{t} A(v).
\end{aligned} \tag{7.27}$$

Therefore, we obtain:

$$B^*(t) = M(t)B(t). \tag{7.28}$$

Now, let us consider a small Δt. Then, we can make the approximation:

$$A^*(t) \approx A(t). \tag{7.29}$$

Hence, considering the definitions of A^* and B^*, we finally obtain:

$$\begin{aligned}
&x^*_{on}(t)\Delta t = M(t)x_{on}(t)\Delta t && \Rightarrow \quad x^*_{on}(t) = M(t)x_{on}(t), \\
&1 - x^*_{on}(t)\Delta t - x^*_{off}(t)\Delta t \approx 1 - x_{on}(t)\Delta t - x_{off}(t)\Delta t && \Rightarrow \quad x^*_{off}(t) = (1-M(t))x_{on}(t) + x_{off}(t).
\end{aligned} \tag{7.30}$$

□

By using the final expression of the modulating function parameters from equation 7.17, we find the values of these parameters, by imposing the additional boundary conditions:

$$\begin{aligned}
0 \le x^*_{on}(t) \le 1, &\qquad 0 \le x^*_{off}(t) \le 1, \\
0 \le r^*_{on}(t) \le 1, &\qquad 0 \le r^*_{off}(t) \le 1,
\end{aligned} \tag{7.31}$$

i.e. the modulated session rates and UE ratios must not exceed the $[0,1]$ interval[4].

In order to satisfy these limits, we choose to slightly adjust, if necessary, the amplitude of the weekly component M_1 because it affects only the weekly variations in the number of online/offline UEs rather than the average. To solve the system of inequalities, let us consider $M(t)$ defined according to equation 7.15, and let us denote t_+ and t_- any t for which we have:

$$\begin{aligned}
\cos(2\pi f_0 t_+ + \phi_M) \ge 0, &\quad \forall\, t_+, \\
\cos(2\pi f_0 t_- + \phi_M) < 0, &\quad \forall\, t_-,
\end{aligned} \tag{7.32}$$

By taking aside each inequality, we obtain the following limits:

[4] According to the definition of the session rate, we cannot have more UEs changing the online or offline state during any infinitesimal Δt than we had before. Likewise, according to the definition of the UE ratios, we cannot have a number of UEs in either online or offline state, greater than the total number of UEs in the system.

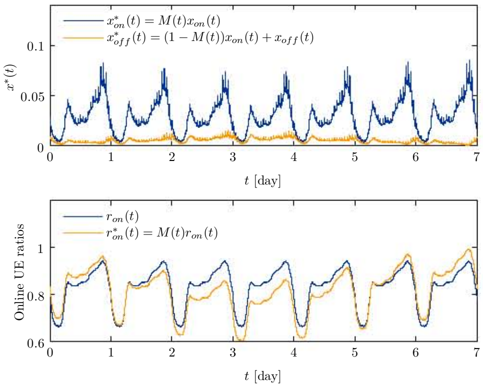


Figure 7.10: The modulated online UE ratio r^*_{on} obtained from x^*_{on} and x^*_{off}.

$$\begin{aligned} -M_0 \leq& M_1 \leq M_0, \\ \max\left\{\frac{1/x_{in}(t_-) - M_0}{\cos(2\pi f_0 t_- + \phi_M)}\right\} \leq& M_1 \leq \min\left\{\frac{1/x_{in}(t_+) - M_0}{\cos(2\pi f_0 t_+ + \phi_M)}\right\}, \\ \max\left\{\frac{1 - M_0 + x_{off}(t_-)/x_{on}(t_-)}{\cos(2\pi f_0 t_- + \phi_M)}\right\} \leq& M_1 \leq \min\left\{\frac{1 - M_0 + x_{off}(t_+)/x_{on}(t_+)}{\cos(2\pi f_0 t_+ + \phi_M)}\right\}, \\ \max\left\{\frac{1 - M_0 - (1 - x_{off}(t_+))/x_{on}(t_+)}{\cos(2\pi f_0 t_+ + \phi_M)}\right\} \leq& M_1 \leq \min\left\{\frac{1 - M_0 - (1 - x_{off}(t_-))/x_{on}(t_-)}{\cos(2\pi f_0 t_- + \phi_M)}\right\}, \\ \max\left\{\frac{1/r_{on}(t_-)}{\cos(2\pi f_0 t_- + \phi_M)}\right\} \leq& M_1 \leq \min\left\{\frac{1/r_{on}(t_+)}{\cos(2\pi f_0 t_+ + \phi_M)}\right\}. \end{aligned} \tag{7.33}$$

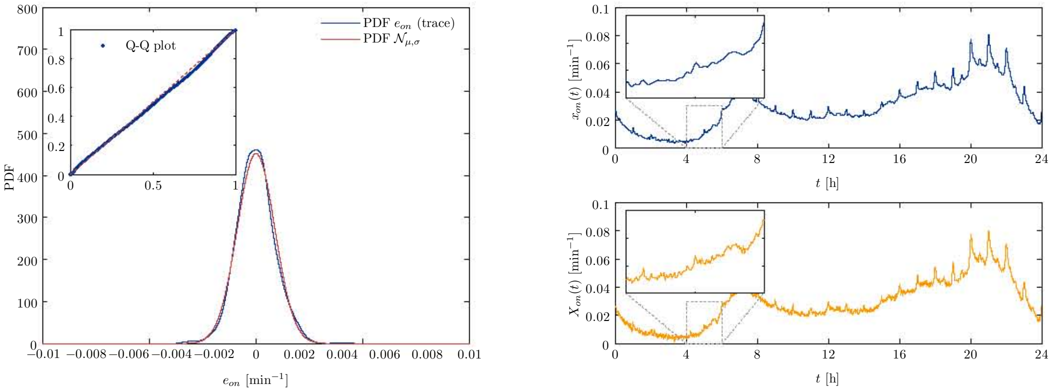
By solving the equation for M_1, subject to the previous constraints, we obtain a value of 0.0741, slightly smaller than the ideal 0.0943 given by the equation 7.17[5]. Therefore, the weekly variations in the number of online/offline UEs will not be as large as the trace, but we believe that choosing the modulation function in this manner represents a good compromise for our model. In the table 7.3, we summarize the parameters of the modulating function $M(t)$.

Parameter	Value
M_0	0.96068
M_1	0.07408
ϕ_M	0.22189

Table 7.3: The parameters of the modulating function $M(t)$.

Finally, we modify the online/offline session rate, such that the resulting online/offline UE ratios are modulated by $M(t)$. In the figure 7.10, we illustrate the results of the previous theorem where we obtained a modulated online UE ratio r^*_{on}, by replacing the session rates with x^*_{on} and x^*_{off}. Finally, we verify that these changes to the session rate does not alter significantly our original frequency-based model. To this end, the cosine similarity is:

[5] In practice, the value selected for M_1 can be slightly lower to count for the error in machine precision.



(a) Comparing the PDF and quantile of the error e_{on} with the normal distribution $\mathcal{N}_{\mu,\sigma}$.

(b) Modeling the online session rate as the x_{on} mean with an additive Gaussian error.

Figure 7.11: Modeling the stochastic properties of the online session rate.

$$
\begin{aligned}
S_{\cos,on} &= \frac{x_{on} \cdot x^*_{on}}{\|x_{on}\|\|x^*_{on}\|} \approxeq 0.9986, \\
S_{\cos,off} &= \frac{x_{off} \cdot x^*_{off}}{\|x_{off}\|\|x^*_{off}\|} \approxeq 0.9479,
\end{aligned}
\tag{7.34}
$$

which indicates that the differences between the original and modified session rates are negligible.

7.2.3 Stochastic Properties

The online, offline and channel session rates, generated with the previous model, are deterministic in nature. The idea behind our analysis and modeling, based on the frequency characteristics, is to exploit the dominant frequency components that generate repetitive patterns from time scales as long as a week down to several minutes. However, as a disadvantage, our approach does not capture any inherent randomness from the real-life process. In reality, while either session rates are not independent random variables, being coupled by both the human activity on large (diurnal, weekly, etc.) scale and the TV program schedule on small (hours, minutes) scale, they are nevertheless random. Therefore, in this section, we analyze in greater depth the available data, toward adding a stochastic property to the session rates. Here, we present the modeling of the stochastic properties of the online session rate x_{on}, since the analysis is similar for the others.

We begin from the premise that, regardless of its lack of randomness, the frequency model represents an average. It is therefore reasonable to conclude that the real online session rate represents a random deviation from this average. Let us denote by $e_{on}(t)$ the error between the trace and model online session rate, i.e.:

$$
e_{on}(t) = x_{on}(t)|_{trace} - x_{on}(t)|_{model}. \tag{7.35}
$$

Our objective is to determine the probability distribution characterizing this error. Intuitively, there are two causes for this error: (i) the approximation in our frequency model that does not capture all relevant frequency components, and (ii) the random nature of users watching TV. Since the first cause is a deterministic property, as it repeats at regular intervals, we want to exclude it, as much as possible from our analysis.

A first observation of the error in time domain, indicates that the error pattern is not similar during different periods of the same day, being characterized by larger variations during evening hours caused by the approximations in the frequency model. In order to find the typical time of day during which the error samples are representative, we compute the error mean and variance at different moments using a sliding window W. To determine a typical mean μ and variance σ^2 for the error sample set, we exclude the evening peak TV usage, since allowing such a large variance would introduce high deviations in the session rate at moments, such as during the night or morning, when normally they would not happen.

On the remaining data set, we apply the kernel density estimation to find its probability density function. As the figure 7.11(a) illustrates, the PDF is similar to a normal distribution having the same μ and σ as the error samples. Therefore, it is reasonable to conclude that the session rate error can be modeled by a Gaussian process $\mathcal{N}_{\mu,\sigma}$ where the mean μ is close to zero. A Q-Q plot, in the same figure, confirms the match between the trace and the normal distribution.

Toward this end, if we denote the online session rate random variable by $X_{on}(t)$, we have:

$$\begin{aligned} X_{on}(t) &= x_{on}(t) + \mathcal{N}_{\mu,\sigma}, \\ X^*_{on}(t) &= x^*_{on}(t) + \mathcal{N}_{\mu,\sigma}, \end{aligned} \tag{7.36}$$

where X^*_{on} and x^*_{on} are the corresponding weekly-modulated versions. Figure 7.11(b) illustrates the difference between the stochastic and deterministic versions of the online session rate. By following the same method, we find the mean μ and variance σ^2 (or standard deviation σ) for the offline and channels sessions, summarizing the resulting values in table 7.4.

Error	μ	σ
e_{on}	$0\,(\sim 10^{-6})$	$8.8219 \cdot 10^{-4}$
e_{off}	$0\,(\sim 10^{-6})$	$2.1437 \cdot 10^{-4}$
e_{ch}	$0\,(\sim 10^{-5})$	0.0061

Table 7.4: The parameters of the normal distributed error e_{on}, e_{off} and e_{ch}.

7.3 Channel Popularity

Qiu et al. did a comprehensive study of the channel popularity in a large IPTV system (Qiu, Ge, Lee, Wang, Zhao, & Xu, 2009). Their analysis focused on how users change between TV channels, and concluded that the popularity of a TV channel is determined by the following user actions, which is typical to similar studies on IPTV (Cha, Rodriguez, Crowcroft, et al., 2008):

- *Target channel switching*, when the user changes directly to a TV channel of interest.
- *Sequential channel switching*, corresponding to the user browsing through the available channels, in search of an interesting TV program.

In practice, the nature of sequential switching depends on whether the user was previously watching a TV channel or its user equipment (i.e. its user equipment) was turned off. When the user resumes watching TV without tuning in to a specific channel, after a period of inactivity, the design of the UE is to resume from the last known channel. Therefore, we can further divide sequential switching in:

- *Turning on* the UE/TV set, where the last known channel is resumed.

Browsing forward $q_sq_cq_f$: 37 %
Browsing backward $q_sq_cq_b$: 15 %

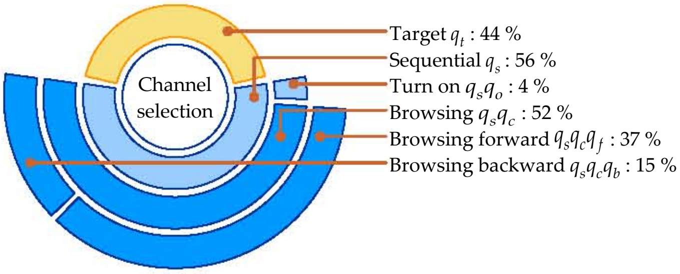


Figure 7.12: Summarizing the types of channel selection.

- *Browsing*, when the user actually changes from an existing channel *forward* to the next or *backward* to the previous one. The results from the study indicate to us that in over 72 % of browsing events, the user forwards to the next channel.

Name	Notation	Value
Target switching	q_t	0.44
Sequential switching	$q_s = 1 - q_t$	0.56
Turn-on sequential switching	q_o	0.08
Browsing sequential switching	$q_c = 1 - q_o$	0.92
Forward browsing sequential switching	q_f	0.72
Backward browsing sequential switching	$q_b = 1 - q_f$	0.28

Table 7.5: Set of probabilities for channel switching events.

We assign a probability q for every type of channel switching event. The table 7.5 and the figure 7.12 summarize the types of channel switching and their corresponding probabilities. In contrast to the sequential switching, where the future channels are dependent on the previous ones, the target switching is a memoryless stochastic process. To this end, we introduce the target switching popularity of a channel i as the probability of switching to that particular channel, denoted by P_i. Through their comprehensive study, Qiu et al. concluded that the popularity of a TV channel has two components:

- A *long-term* component, P_μ, that follows a *zipf-exponential* distribution.
- A *instantaneous* component, P_t, that follows a *mean reversion* model.

Since we find their model a good one, we shall reuse the foundation of their work in the implementation of our workload. Therefore, in the following two sections we describe the mathematical background as presented in (Qiu, Ge, Lee, Wang, Zhao, & Xu, 2009), while we extend some equations, such as the solution of the instantaneous probability, to a more general form.

7.3.1 Long-Term Popularity for Target Switching

Let us consider an IPTV system with N channels, in their descending order of their popularity. The long term popularity of channel i is:

$$P_{\mu,i} = \begin{cases} \dfrac{Q_1}{Q_0 i^{\alpha'}} & i < N' = N/10, \\ \dfrac{e^{-\beta i + Q_2}}{Q_0} & \text{otherwise,} \end{cases} \tag{7.37}$$

where the popularity of only the top ten percentile of the channels, denoted by N', is *zipf*, while for the rest is an *exponential* (Qiu, Ge, Lee, Wang, Xu, & Zhao, 2009). For Q_2, in the interest of accuracy, we impose:

$$Q_2 = \ln\left(Q_1 N'^{-\alpha'}\right) + N'\beta, \tag{7.38}$$

and Q_0 is a normalizing factor such that:

$$\sum_{i=1}^{N} P_{\mu,i} = 1. \tag{7.39}$$

Parameter	Value	Parameter	Value	Parameter	Value
N	700	α	0.45	β'	0.006
N'	70	α'	0.513	Q_1	12.642

Table 7.6: The parameters of the mean channel popularity model for target switching.

7.3.2 Instantaneous Popularity for Target Switching

The instantaneous popularity of a TV channel with long-term popularity P_μ is modeled by a Ornstein-Uhlenbeck stochastic process described by the stochastic differential equation:

$$\mathrm{d}P_t = \theta\left(P_\mu - P_t\right)\mathrm{d}t + \sigma \mathrm{d}W_t, \tag{7.40}$$

where we have:

- θ is the *mean reversion rate*,
- μ is the *long-term mean*,
- σ is the *volatility* of the Weiner process,
- W_t is a *Weiner process* characterized by $W_t - W_s \sim \mathcal{N}_{0,t-s}$.

While Qiu et al. offer a solution at discrete moments of time, our objective is to solve the stochastic differential equation for any real time moment t, when a subscriber changes to a new channel.

Theorem 3. *Let t be moment of the last channel change, and Δt the duration to a subsequent change to the same channel, by the same or a different user. Then, the instantaneous channel popularity at time $t + \Delta t$ is:*

$$P_{t+\Delta t} = P_t e^{-\theta\Delta t} + P_\mu\left(1 - e^{-\theta\Delta t}\right) + \sigma\sqrt{\frac{1 - e^{-2\theta\Delta t}}{2\theta}}\mathcal{N}_{0,1}, \tag{7.41}$$

where $\mathcal{N}_{0,1}$ is a random number with standard normal distribution.

Proof. We solve the stochastic differential equation:

$$\frac{\mathrm{d}P_t}{\mathrm{d}t} + \theta P_t = \theta P_\mu + \sigma \mathrm{d}W_t. \tag{7.42}$$

By multiplying both sides with $\mathrm{e}^{\theta t}$, we obtain:

$$\frac{\mathrm{d}P_t}{\mathrm{d}t}\mathrm{e}^{\theta t} + \theta P_t \mathrm{e}^{\theta t} = \theta P_\mu \mathrm{e}^{\theta t} + \sigma \mathrm{e}^{\theta t}\mathrm{d}W_t. \tag{7.43}$$

Then, by integrating over the interval $[t, t + \Delta t]$:

$$\begin{aligned} P_t e^{\theta t}\Big|_t^{t+\Delta t} &= P_\mu \, e^{\theta t}\Big|_t^{t+\Delta t} + \sigma \int_t^{t+\Delta t} e^{\theta t} dW_t, \\ P_{t+\Delta t} e^{\theta(t+\Delta t)} - P_t e^{\theta t} &= P_\mu \left(e^{\theta(t+\Delta t)} - e^{\theta t}\right) + \sigma \int_t^{t+\Delta t} e^{\theta t} dW_t \, . \end{aligned} \tag{7.44}$$

In the previous equation, the last term of the right-hand side is the Itō's integral. By dividing both sides to $e^{\theta(t+\Delta t)}$, and considering the solution to Itō's integral (Dixit et al., 1994), we have completed our proof.

□

Considering an initial channel popularity P_0, at time t the channel popularity P_t has a *normal distribution* with mean and variance (Kloeden et al., 1994):

$$\mathrm{E}\left[P_t\right] = P_0 e^{-\theta t} + P_\mu \left(1 - e^{-\theta t}\right), \tag{7.45}$$

$$\mathrm{Var}\left[P_t\right] = \frac{\sigma^2 \left(1 - e^{-2\theta t}\right)}{2\theta}. \tag{7.46}$$

The parameters of the model are given in the table 7.7, while for σ we have:

$$\sigma = P_u \cdot c_v \sqrt{2\theta}. \tag{7.47}$$

Parameter	Value
θ	0.12
c_v	0.5

Table 7.7: The parameters of the instantaneous channel popularity model for target switching.

7.3.3 Channel Ordering

The model of the target switching popularity assumes the TV channels are ordered by descending popularity. However, the order of channels set, as broadcasted to the users, is usually different with TV channels being sorted around their inclusion in a certain TV package such as basic, kids & music, adventure, sports, pay-per-view, etcetera. Subscribers may also customize the order of the channels according to their preference.

Since the TV channels order plays an important role for the sequential switching, it becomes necessary to establish a channel order that considers both the broadcasting and user customization. Toward this end, Qiu et al. propose a simple permutation function, with the acceptance of a slightly higher error if the channels received by users are highly clustered.

Let us denote by σ the uniform permutation function for a given user, such that a TV channel on the position i in the decreasing popularity set, will be tuned on position $\sigma(i)$ at the user equipment/TV set. The permutation function is bijective, and therefore there exists an inverse $\sigma^{-1}(i)$, such that a channel on the position i in the received set will have the position $\sigma^{-1}(i)$ in the ordered set.

7.4 Conclusions

The design of the P2PTV-AS session and peer coordination algorithms requires a thorough understanding of the user activity in a large commercial IPTV system. Unfortunately, due to the closed nature of the existing deployments, to date there are only a few comprehensive studies on how residential viewers watch TV. The work by Qiu et al. is one of the few that present a comprehensive data set, analyzed from a number of dimensions (Qiu, Ge, Lee, Wang, Xu, & Zhao, 2009; Qiu, Ge, Lee, Wang, Zhao, & Xu, 2009). These include the process of the user equipment online, offline and channel sessions; the channel popularity and its dynamics; users classification and channel preference. By relying on their data, the authors propose a simple IPTV workload generator called *Simulwatch*.

Although the work of Qiu el al. represents an invaluable foundation, our analysis of their model and generator finds it inadequate for our purpose. Mainly, the Simulwatch algorithms are incomplete on certain aspects of the workload characteristics, like the number of online users and the activity dynamics on larger time-scales, that are essential for a good design of the P2PTV system. In addition, it suffers a few flaws, which, in our opinion, do not model correctly the real-life data.

In order to address these issues, we extended their work by proposing a new session rate model, based on dominant spectral components, rather than a continuous frequency characteristic. We prove that the session rate is linked to the number of user equipments online at any given time, and that the steady-state of the system does not depend on the initial conditions. Our workload includes the weekly pattern of the user activity, by using a modulating functions that alters the session rate, and indirectly the number of online UEs. Unlike Simulwatch, we do not leave the session rate entirely deterministic, by measuring and adding a normally-distributed stochastic component.

Chapter 8

Workload Synthesis

8.1 Overview

While the analysis and modeling of the IPTV workload is relatively straightforward, synthesizing it becomes more challenging. Since the workload model is an approximation of the reality, and due to the subtle differences and incomplete data covering all details of how users watch TV, the different model parameters do not match perfectly. Toward this end, this chapter describes the details of generating an IPTV workload starting from our model.

The figure 8.1 illustrates that our synthesis algorithm divides the implementation of the workload model into the following logical steps:

- The user activity *session rate*, by generating the starting moments for online, offline and channel sessions.
- The user activity *online and offline session length*, by generating the sequence of online/offline

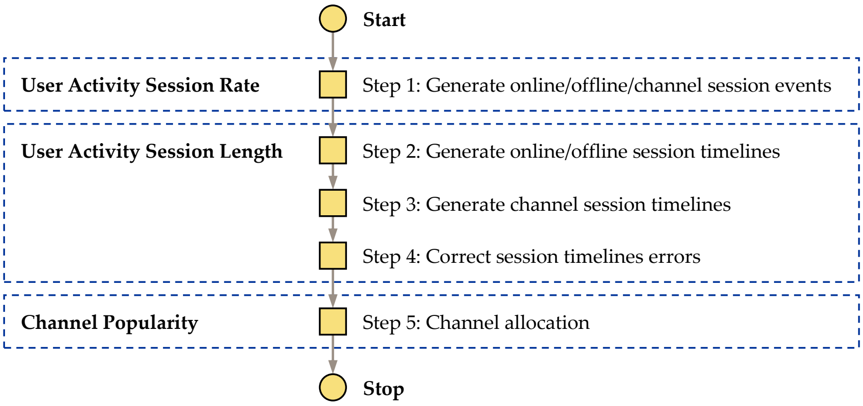

Start
User Activity Session Rate
Step 1: Generate online/offline/channel session events
User Activity Session Length
Step 2: Generate online/offline session timelines
Step 3: Generate channel session timelines
Step 4: Correct session timelines errors
Channel Popularity
Step 5: Channel allocation
Stop

Figure 8.1: Overview of the workload synthesis algorithm.

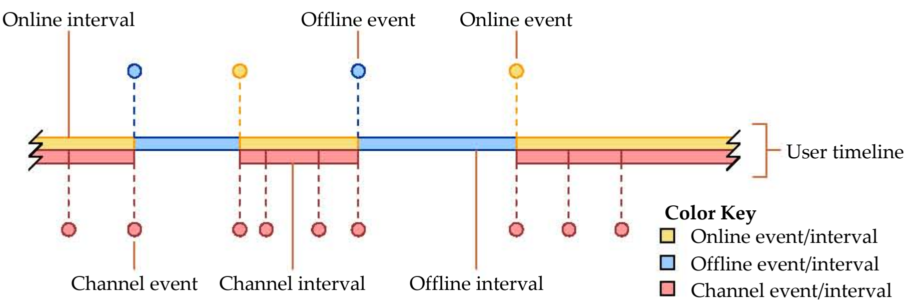


Figure 8.2: Definitions of elements used in the workload algorithm.

sessions for the users in the system.

- The user activity *channel session length*, by generating the sequence of channel sessions for the users in the system.
- The *channel popularity*, by allocating channels to the existing channel sessions.

8.2 User Activity

In order to keep the implementation simple, the synthesis of the user activity separates the implementation of the *session rate* and *session length*. This approach has the advantage of efficiency, as the computing cost is greatly reduced, especially for a large number of users.

8.2.1 Model Analysis

The online, offline and channel session states of the user activity must accommodate two model distributions: (i) the *session rate* and (ii) the *session duration*. We remind to the reader that the session rates are the random variables X^*_{on}, X^*_{off} and X_{ch}, generated using the daily frequency model, the weekly modulating function and the normally-distributed stochastic component. The session durations d_{on}, d_{off} and d_{ch} are generated from the fitted hyper-exponential distributions.

To simplify the understanding of the workload algorithm, we define the following elements, visualized in the figure 8.2:

- A session *event* is the time moment of the beginning of a given session. Depending on the session type, we have online, offline and channel session events.
- A session *interval* is the time duration spanned by a given session. Similar to the session event, we have online, offline and channel session intervals.
- A session *timeline* is the sequence of sessions a given user generates through its TV watching activity.

Our objective is to generate, for every user, a set of events representing the time moments when the user's UE changes its online, offline and/or channel state, given the following constraints:

(i) For an individual user, there are alternating the online and offline session intervals.

(ii) For an individual user, the channel session intervals must overlap only with the online session intervals.

(iii) The duration of every online, offline and channel interval is d_{on}, d_{off} and d_{ch}, respectively.

(iv) When all users are taken together, the superposition of their online, offline and channels events determines a process having the rate X^*_{on}, X^*_{off} and X_{ch}, respectively.

(v) The online and offline events are a subset of the channel events, as turning the UE on are off equally involves a channel state change.

In essence, our objective is to match the session events having X^*_{on}, X^*_{off} and X_{ch} rate with the session intervals of d_{on}, d_{off} and d_{ch} duration, while maintaining a constant number of total users in the system. We begin with an analysis of the user activity model to determine whether the above constraints can be met, and if not, what adjustments are necessary. Specifically, we verify if the number of online/offline session events accommodate the session intervals.

8.2.1.1 Online/Offline Sessions

Given the online/offline session interval duration d_{on} and d_{off} following a hyper-exponential distribution, each with n parameters p_i and λ_i, as shown in the table 7.1, the mean or expected value for each duration is:

$$\begin{aligned} \bar{d}_{on} &= \sum_{i=1}^{n} \frac{p_i}{\lambda_i}\bigg|_{on}, \\ \bar{d}_{off} &= \sum_{i=1}^{n} \frac{p_i}{\lambda_i}\bigg|_{off}. \end{aligned} \tag{8.1}$$

Let us consider an observation interval t_{max}, such that $t_{max} \to \infty$. For a given user equipment, the mean number of online and offline intervals that can be accommodated during t_{max} is:

$$\bar{n}_{d,o} = \frac{2t_{max}}{\bar{d}_{on} + \bar{d}_{off}} \approxeq 1.02 \cdot 10^{-4} t_{max}. \tag{8.2}$$

On the other hand, the online/offline session events have the average rate $x^*_{on}(t)$ and $x^*_{off}(t)$ normalized to the number of offline and online users at each time moment t, respectively. Give the fraction of online/offline UEs $r^*_{on}(t)$ and $r^*_{off}(t)$, we can write the absolute on/off session rate for an individual UE as[1]:

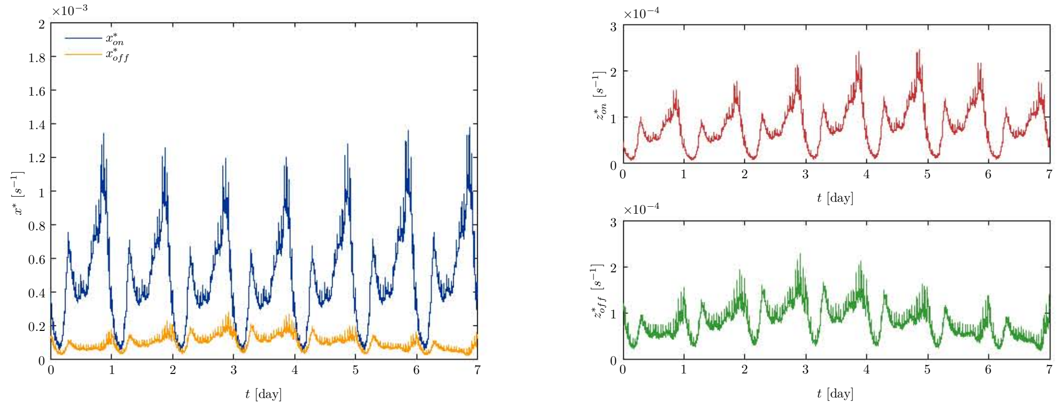
$$\begin{aligned} z^*_{on}(t) &= x^*_{on}(t) r^*_{off}(t), \\ z^*_{off}(t) &= x^*_{off}(t) r^*_{on}(t). \end{aligned} \tag{8.3}$$

The figure 8.2.1.1 illustrates the difference between the two forms of the session rate. One may observe that while the normalized online/offline rates differ by an order of magnitude, the absolute forms are nearly identical. This result is expected since the total number of users/UEs in the system remains constant over time. Therefore, in average, the number of UEs switching on should be equal to the number of UEs switching off.

The average session rate for one UE during t_{max} is:

$$\begin{aligned} \bar{z}^*_{on} &= \frac{1}{t_{max}} \int_0^{t_{max}} z^*_{on}(t)\, \mathrm{d}t, \\ \bar{z}^*_{off} &= \frac{1}{t_{max}} \int_0^{t_{max}} z^*_{off}(t)\, \mathrm{d}t. \end{aligned} \tag{8.4}$$

[1] We use the notation z^* for the weekly-modulated version of the absolute session rate, in order to differentiate it from the non-modulated version, denoted by z.



(a) Normalized online/offline session rate x^*_{on}, x^*_{off}.

(b) Absolute online/offline session rate z^*_{on}, z^*_{off} for one UE.

Figure 8.3: Comparing the normalized online/offline session rate with the absolute version.

Considering the model expressions for the session rates, we can compute the average number of online/offline session events a UE experiences during t_{max}:

$$\bar{n}_{z,o} = \bar{n}_{z,on} + \bar{n}_{z,off} = \int_0^{t_{max}} \left(z^*_{on}(t) + z^*_{off}(t) \right) \mathrm{d}t \approxeq 1.55 \cdot 10^{-4} t_{max}. \tag{8.5}$$

From the last result, we obtain:

$$\frac{\bar{n}_{d,o}}{\bar{n}_{z,o}} \approxeq 0.65. \tag{8.6}$$

This fraction shows that, in average, the number of online/offline session intervals that can be accommodated by a UE during an arbitrary large period is approximately only 65 % of the number of online/offline events that have to be generated by the UE during the same time. Essentially, the average length of the sessions is larger than the average distance between the online/offline session events. This inconsistency originates in the limited data that is available for the session rate and duration, i.e. one day, compared to our model that we extended to weekly time scales.

In order to accommodate this difference, we implement the workload algorithm to prioritize the session rate (x^*_{on}, x^*_{off}) over the session interval duration (d_{on}, d_{off}). We justify our approach by considering the following aspects. The session rate includes high transients, i.e. flash crowds, at specific moments of time such as hour boundaries, corresponding to the TV program schedule. The nature of these changes in the session rate makes it difficult to approximate with a good precision. For the design of a P2PTV algorithms, the online/offline session rate is analogous to the churn rate. We expect that the performance of such a peer-to-peer system is more sensitive to peers arriving or leaving the system in a short period of time, rather than to small changes in the duration of peers staying online or offline. Finally, we show that the approximated session interval duration can be sufficiently close to the model.

8.2.1.2 Channel Sessions

We apply the same principle to the channel sessions. Given the hyper-exponential distribution mode, the mean duration of a channel interval is:

$$\bar{d}_{ch} = \sum_{i=1}^{n} \frac{p_i}{\lambda_i}\bigg|_{ch}, \tag{8.7}$$

where p_i and λ_i are the model parameters from the table 7.1. During an observation interval of duration t_{max}, the expected number of channel intervals is:

$$\bar{n}_{d,c} = \frac{t_{max}}{\bar{d}_{ch}} \approxeq 0.0012. \tag{8.8}$$

Similarly to the online-offline sessions, the channel events rate at the moment t is $x_{ch}(t)$, normalized to the number of online users, and we can write the absolute events rate as:

$$z_{ch}(t) = x_{ch}(t) r^*_{on}(t). \tag{8.9}$$

The mean number of channel events, during the same arbitrary interval t_{max}, is:

$$n_{z,c} = \int_0^{t_{max}} z_{ch}(t)\, \mathrm{d}t \approxeq 8.25 \cdot 10^{-4}, \tag{8.10}$$

and we obtain:

$$\frac{n_{d,c}}{n_{z,c}} \approxeq 1.39. \tag{8.11}$$

The last result shows that the number of intervals accommodated by the observation interval based on their length is slightly higher than the number of session events. For the same reasons as before, in our implementation, we give priority to the channel events rate, which results in a mean interval duration higher than the model. In the next chapter, we analyze this difference in greater detail, and we show that it is a reasonable compromise.

8.2.2 Session Events

Our previous analysis shows that the absolute channel event rate exceeds the online/offline rate by an order of magnitude. Therefore, the idea behind our algorithm is to generate the sequence of channel events first, and subsequently interpolate the online/offline events. These events will then be *linked* to the nearest channel events such as an online event t_{on} is anterior to its linked channel event t_{ch}, and an offline event t_{off} is posterior to its linked channel event t_{ch}.

The figure 8.4 illustrates the generation of session events. The algorithm initializes the moment of the first online, offline and channel event, and the number of online and offline users/UEs. Considering a fixed number of U subscribers to the IPTV service, the latter can take the form:

$$\begin{aligned} u_{on} &= \delta_U U, \\ u_{off} &= (1 - \delta_U) U. \end{aligned} \tag{8.12}$$

The parameter δ_U represents the initial fraction of online UEs. Since the algorithm ensures the convergence of the workload after a reasonable elapsed time, δ_U can be set to any initial value in the interval $[0, 1]$. However, we can minimize the convergence interval if we choose $\delta_U = 0.8117$. The main loop of the algorithm generates a new channel event per iteration, until the maximum time is reached. At every iteration, we check whether the last online or offline event should be linked to the last channel event. When any of these events are linked, we add them to the timeline, and then generate a new one.

Online/offline events corresponding to the users turning their UE on and off, are linked automatically to the nearest channel events. This approach maintains the appropriate session rate distribution for all events, while making the logical connection that turning on the UE is followed at the same time by a tune in to the last known TV channel. Similarly, turning off the UE should be preceded by a tune out from the current TV channel. The *link type* distinguishes between these channel events. Because two linked events do not occur the same time, there exists a short

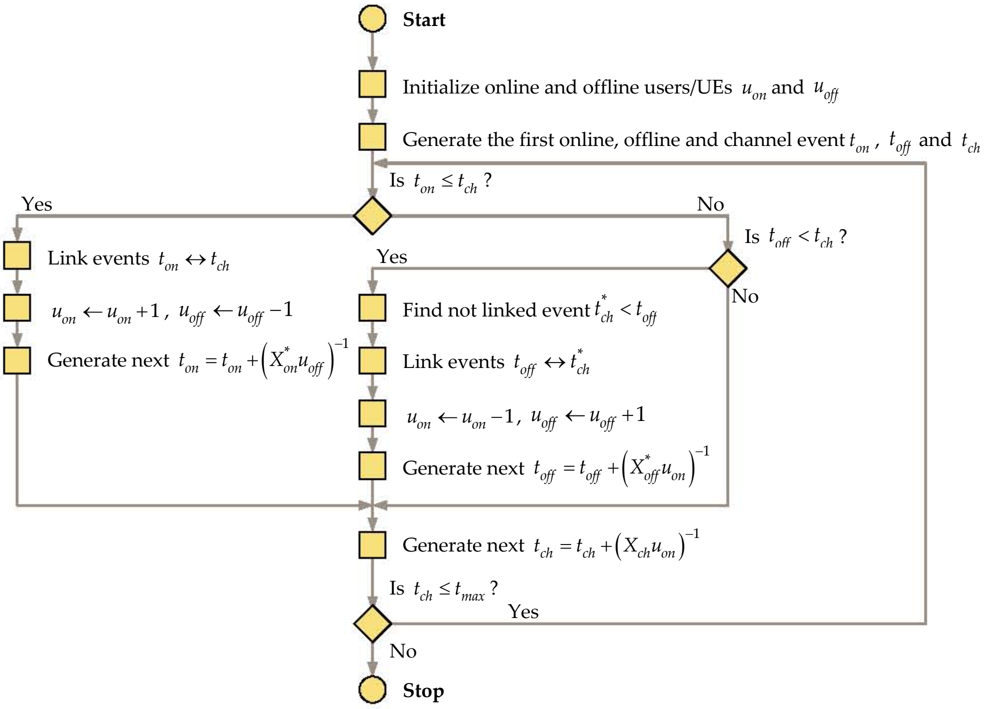

Start
Initialize online and offline users/UEs u_{on} and u_{off}
Generate the first online, offline and channel event t_{on}, t_{off} and t_{ch}
Is $t_{on} \le t_{ch}$?
Yes
No
Is $t_{off} < t_{ch}$?
Yes
No
Link events $t_{on} \leftrightarrow t_{ch}$
$u_{on} \leftarrow u_{on} + 1$, $u_{off} \leftarrow u_{off} - 1$
Generate next $t_{on} = t_{on} + \left(X_{on}^{*} u_{off}\right)^{-1}$
Find not linked event $t_{ch}^{*} < t_{off}$
Link events $t_{off} \leftrightarrow t_{ch}^{*}$
$u_{on} \leftarrow u_{on} - 1$, $u_{off} \leftarrow u_{off} + 1$
Generate next $t_{off} = t_{off} + \left(X_{off}^{*} u_{on}\right)^{-1}$
Generate next $t_{ch} = t_{ch} + \left(X_{ch} u_{on}\right)^{-1}$
Is $t_{ch} \le t_{max}$?
Yes
No
Stop

Figure 8.4: The algorithm for generating the session events.

undefined period, usually of a few milliseconds, when the UE may be online but not watching any TV channel. While the nature of this error is purely probabilistic, in practice we consider it a negligible error.

8.2.3 Online/Offline Session Intervals

Following the generation of the session events in the sets $\mathcal{E}_{on}$, $\mathcal{E}_{off}$, and $\mathcal{E}_{ch}$, we include the session intervals of duration d_{on} and d_{off}, organized in the sets $\mathcal{I}_{on}$ and $\mathcal{I}_{off}$. Unlike the session events, which are independent between them given their rate, computing the session intervals is subject to a number of constraints. First, an online/offline session interval originates in an online/offline event and terminates at an offline/online event. Second, the number of online and offline intervals that overlap at any given time must equal to the number of users/UEs. Finally, according to the

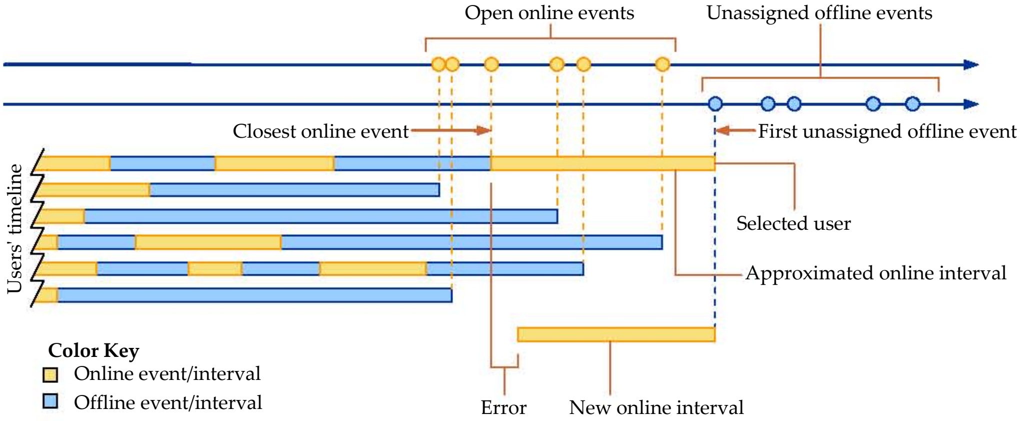


Figure 8.5: Generating the online/offline intervals.

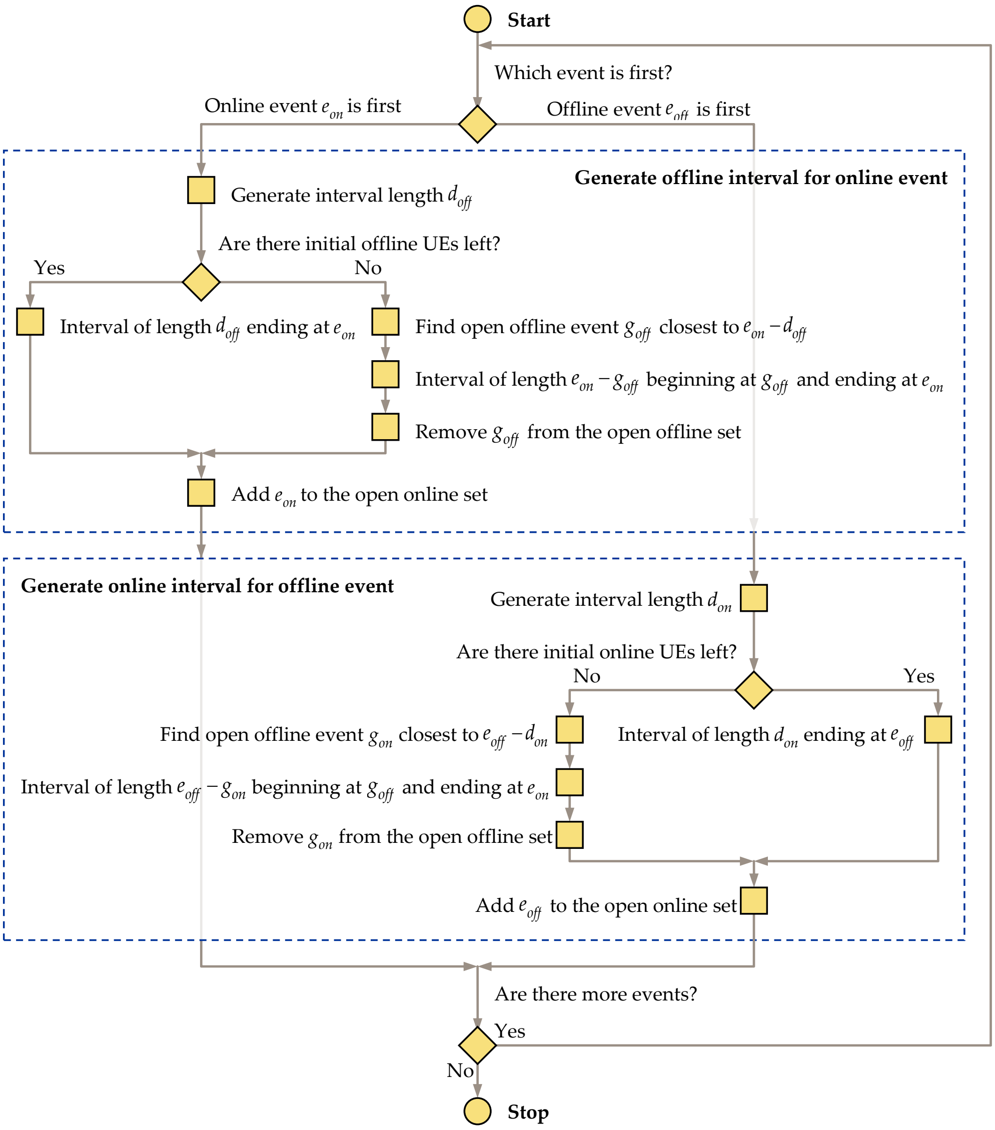

Start
Which event is first?
Online event e_{on} is first
Offline event e_{off} is first
Generate offline interval for online event
Generate interval length d_{off}
Are there initial offline UEs left?
Yes
No
Interval of length d_{off} ending at e_{on}
Find open offline event g_{off} closest to $e_{on} - d_{off}$
Interval of length $e_{on} - g_{off}$ beginning at g_{off} and ending at e_{on}
Remove g_{off} from the open offline set
Add e_{on} to the open online set
Generate online interval for offline event
Generate interval length d_{on}
Are there initial online UEs left?
No
Yes
Find open offline event g_{on} closest to $e_{off} - d_{on}$
Interval of length d_{on} ending at e_{off}
Interval of length $e_{off} - g_{on}$ beginning at g_{off} and ending at e_{on}
Remove g_{on} from the open offline set
Add e_{off} to the open online set
Are there more events?
Yes
No
Stop

Figure 8.6: The algorithm for generating the online/offline session intervals.

objectives we have set for our implementation, we must include all events at a tradeoff with an approximate session length.

To achieve these constraints, we employ a dynamic programming approach where we fix the users/UEs number constraint and we allocate unassigned session events to the user *timelines*. A user timeline represents the repetitive sequence of online-offline events and intervals associated to each UE. Every new interval in the timeline is assigned such that we keep the session length error at a minimum.

We maintain two sets $\mathcal{O}_{on}$ and $\mathcal{O}_{off}$, storing the beginning events for the current open session intervals. These are sessions that have a beginning event, but we have not determined an ending event, and their length is as of yet unknown. Ideally, the sets $\mathcal{O}_{on}$ and $\mathcal{O}_{off}$ can be ordered such that searching is done in logarithmic time, while inserting new elements at the end is done an amortized constant time.

By parsing the event sets $\mathcal{E}_{on}$, $\mathcal{E}_{off}$, from the smallest to the largest, for every new online/offline unassigned event e_{on} or e_{off} we generate a new *closed* offline/online interval of length d_{off} or d_{on}, having the known hyper-exponential distribution. Then, we search for the event in the $\mathcal{O}_{off}$ or

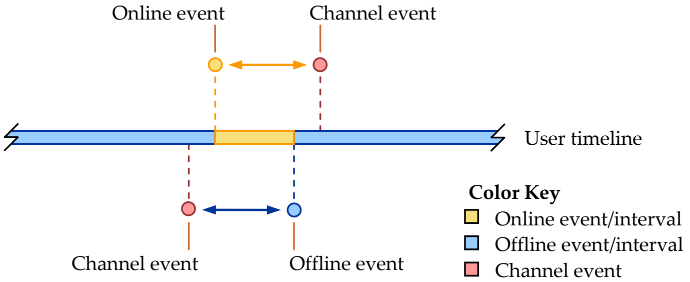
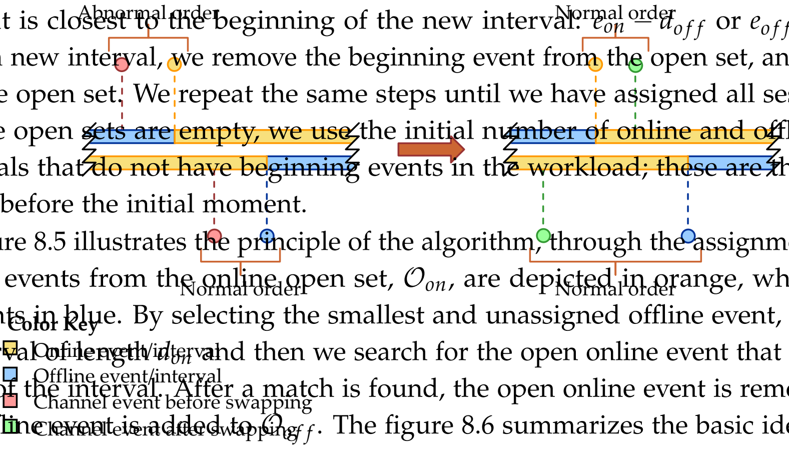


Figure 8.7: Exception situation that requires a correction to the session timelines.

$\mathcal{O}_{on}$ set that is closest to the beginning of the new interval: $e_{on} - d_{off}$ or $e_{off} - d_{on}$. After a we generated a new interval, we remove the beginning event from the open set, and insert the ending event to the open set. We repeat the same steps until we have assigned all session events. Since initially the open sets are empty, we use the initial number of online and offline users to create *open* intervals that do not have beginning events in the workload; these are the session intervals that began before the initial moment.

The figure 8.5 illustrates the principle of the algorithm, through the assignment of a new offline event. The events from the online open set, $\mathcal{O}_{on}$, are depicted in orange, while the unassigned offline events in blue. By selecting the smallest and unassigned offline event, we generate a new online interval of length d_{on} and then we search for the open online event that is the closest to the beginning of the interval. After a match is found, the open online event is removed from $\mathcal{O}_{on}$ and the new offline event is added to $\mathcal{O}_{off}$. The figure 8.6 summarizes the basic idea of the algorithm.



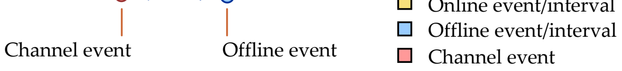
For a session interval of a given length it is unlikely that we find a perfectly matching beginning event. Therefore, the resulting workload will have online/offline session lengths that are close but not identical to our model. Since the online/offline session rate supersedes the online/offline session length, as indicated by our previous our previous analysis, we expect that the average session length to be smaller than the value resulted from the model, in order to accommodate all session events.

8.2.4 Correction for the Session Timelines

Unlike the real-life user activity, the workload synthesis uses the model probability distributions to generate the user events of switching on/off their equipments, and changing their TV channel. Because these events are governed by the laws of random numbers, it is possible, under very improbable circumstances, that two online/offline session events are closer than their corresponding channel events. The figure 8.7 illustrates precisely this situation, where the channels events corresponding to the turning on/off the UE lie outside the corresponding online/offline events.

In the rare situations when this exception occurs, we can take two different approaches. First, we search for an adjacent channel event that most probably is assigned to another user's timeline. If one is found and a swap is possible, by not altering the events order for the other user, we exchange the two channel events. The figure 8.8 shows the principle involved. Generally, when a event swapping occurs, it is done at the compromise of a larger undefined period between the online/offline and the channel events. During the workload synthesis, we may select an upper threshold for the undefined period, which we find acceptable, and only swap events when this threshold is not exceeded.

When swapping is not possible, we do not link any channel events to the online and offline events. In practice, this could correspond to a situation where the user briefly switches on its UE before turning it off again, and during this interval the UE does not tune to any TV channel. In practice, the fraction of online sessions requiring adjustment is very small. In our experience, it



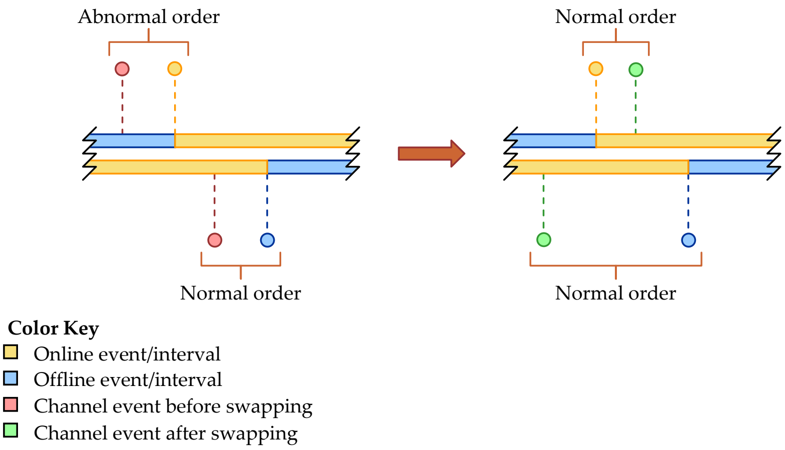


Figure 8.8: Correcting session timelines by swapping the channel events with another user.

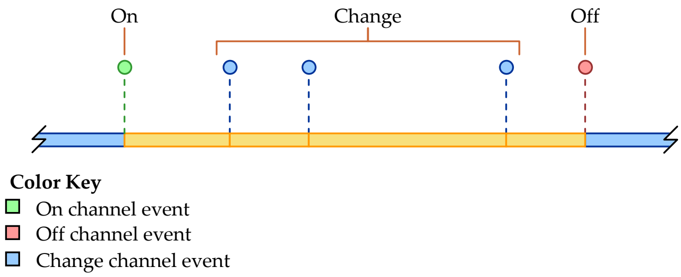


Figure 8.9: The types of channel session events, and their mapping to the channel intervals.

was around 0.5 %, depending on the parameters used for the workload generation, such as the number of users and the duration.

8.2.5 Channel Sessions Intervals

Every channel event is characterized by a type, specifying whether it corresponds to switching the UE *on* or *off*, or a channel *change*. *On* events can only begin channel intervals, *off* events can only end channel intervals, while *change* events can both begin and end them. The figure 8.9 illustrates this mapping for the online interval of a selected UE. The channel *on* and *off* events are associated to UE timelines through their mapping to the online/offline session events. Hence, the current objective is to add to each UE timeline the change events, such that the resulting intervals have a duration that is hyper-exponentially distributed.

The generation of the channel session intervals, denoted by $\mathcal{I}_{ch}$, follows a procedure similar to the online/offline session, based on a dynamic-programming approach. We maintain a set, $\mathcal{O}_{ch}$, of open channel events that can be assigned to the beginning of new intervals. These are the events that have been assigned last to the timeline of each user, and only *on* and *change* events may be added to this set. The channel *off* events occur right before the UE is switched off, and therefore cannot start a new channel interval.

By iterating through all channel events, in chronological order, depending on the event type, we have the following possibilities.

- The channel *on* events are added to the open set, without generating any new interval since the corresponding UE has been offline until the event occurred.

- For the channel *change* events, we generate a new channel interval with a hyper-exponentially

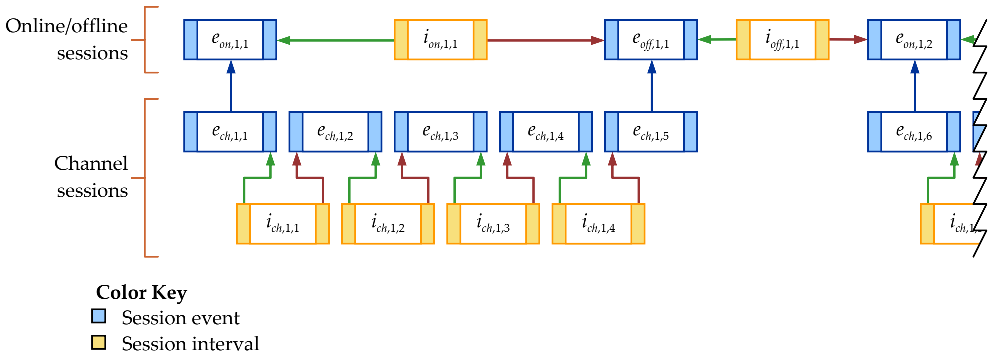


Figure 8.10: The session timeline for an individual user.

distributed duration, and we match it either to the remaining initial UEs, which are the UEs without a previous event on their timeline, or to the open event that is the closest match to the interval duration.

- The events *off* generate a channel interval, which can be either open-ended, if the corresponding UE is an initial one, or are matched to the last event, otherwise. In this last situation, the interval duration does not follow the model. However, we regard this approximation as a good compromise between the workload accuracy and the algorithm efficiency. An in-depth evaluation of the synthetic workload, in the next chapter, shows that the deviations from the model are small enough to be acceptable in practical circumstances.

The figure 8.10 emphasizes the relationships between events and intervals, which allow an easy access to all session components that share the same user timeline. The figure shows the overlapping nature of online/offline and channel session, and the role of the mapping between their events.

8.3 Channel Assignment

At the last step of the workload synthesis, we assign individual TV channels to the user channel sessions. The channel assignment uses the global channel popularity, presented in the section 7.3, consisting of sequential and target switching. The sequential switching uses the memory of the past channel, while the target switching considers the instantaneous popularity of all TV channels. The model assumes that all subscribers have access to the entire set of TV channels, and the order of the channels is randomly shuffled at for viewer.

The assignment algorithm, illustrated in the figure 8.11, is extremely straightforward. For every channel interval, we determine the user action based on the channel change probabilities p_s, p_o, p_f for sequential, UE turn-on, and forward switching, respectively. We do not implement the p_o probability directly, instead it is derived from the type of the channel change events. The most expensive operation, in terms of the required processing power, is the computation of the target switching channel, because it requires updating the instantaneous popularity of all TV channels during every channel change.

8.4 Conclusions

The synthesis of the IPTV workload is a challenging operation. The main reasons are the simplifications behind the workload model and the quasi-independence between the different aspects of the model. These are necessary either because we want to keep the number of model

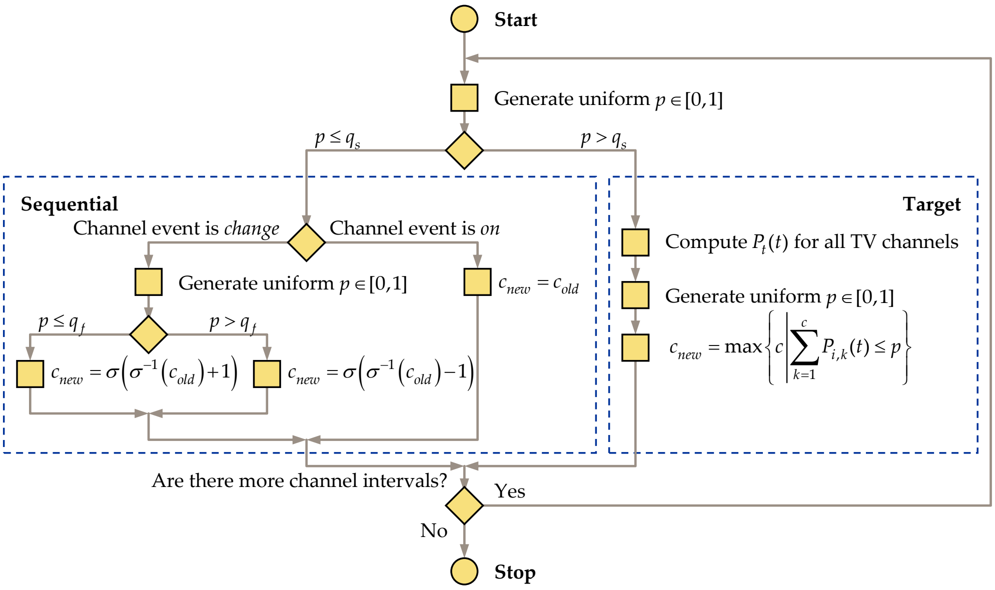

Start
Generate uniform $p \in [0,1]$
$p \le q_s$
$p > q_s$
Sequential
Channel event is *change*
Channel event is *on*
Generate uniform $p \in [0,1]$
$c_{new} = c_{old}$
$p \le q_f$
$p > q_f$
$c_{new} = \sigma\left(\sigma^{-1}\left(c_{old}\right)+1\right)$
$c_{new} = \sigma\left(\sigma^{-1}\left(c_{old}\right)-1\right)$
Target
Compute $P_t(t)$ for all TV channels
Generate uniform $p \in [0,1]$
$c_{new} = \max\left\{ c \,\middle|\, \sum_{k=1}^{c} P_{i,k}(t) \le p \right\}$
Are there more channel intervals?
Yes
No
Stop

Figure 8.11: The algorithm for channel assignment.

parameters small, because we do not have sufficient data to estimate certain parameters with a high degree of confidence, or to keep the synthesis algorithm more efficient. To this end, we propose taking a five-steps approach, where we generate in order each of the following components of the user activity: (i) the session events; (ii) the online/offline session timelines; (iii) the channel session timelines; (iv) the matching between online/offline and channel timelines, and, finally; (v) the session TV channels.

Because the workload model is an approximation of the reality, it is difficult to generate with a perfect match the different aspects of the user behavior. For these reasons, we argue that ceratin compromises are necessary, as long as the error does not have a significant impact. According to our analytical analysis, the first of these appears between the session rate and session duration, which we reconcile by prioritizing the rate over the duration.

The establishing of the user timelines, that is the association between a subset of session events with a certain user, considers the logic flow of the user activity: users turn on their TV set/set-top box UE, change channels and then turn off their equipment. To compute the user timelines under the constraints of the workload model, while providing a high level of scalability in both time and user space, we resort to a dynamic programming approach. The principle of the method consists in remembering the last event from each timeline, and adding the new events to the user timeline that offers the best match for the session duration.

Chapter 9

Workload Evaluation

9.1 Overview

The limited availability of real-life data on the TV watching activity has been the main motivating reason behind our work on workload modeling and synthesis. In addition, a synthetic input has the benefit of allowing us to tune certain parameters, such as the number of TV subscribers or the number of TV channels. By changing the scenario parameters, we are able to test our P2PTV algorithms under a variety of conditions, and therefore we are not limited by the characteristics of the original data set.

Many authors have recognized the importance of system modeling, including Qiu et al., on whose data we based our synthetic approach, and whose model and Simulwatch generator we extended in our current work (Qiu, Ge, Lee, Wang, Xu, & Zhao, 2009; Qiu, Ge, Lee, Wang, Zhao, & Xu, 2009). Especially, when the complete data cannot be released, due to non-disclosure agreements or privacy issues, a good approximation of the real-world provides an invaluable tool for system design and optimization. The scientific literature contains similar publications on the modeling of Internet TV, using data from the PPLive, now named PPTV, service[1] (Vu et al., 2007), or on video-on-demand services (Tang et al., 2003).

From our perspective, the synthetic IPTV workload has a paramount importance on the design of the P2PTV application server. The core idea of the session and peer coordination algorithms, SCA and PCA, is to anticipate the user demand and take a preemptive corrective action. Hence, to enable a proper design and testing, the user demand must reflect real-working conditions. To this end, in this chapter, we analyze the synthetic workload against the available data, to determine whether our extensions to the Simulwatch model, and the approximations made in the implementation, are acceptable.

We approach the evaluation of the IPTV workload from both a narrow and a broad perspective. In the first case, we verify the characteristics of the synthetic user activity against the measurement data that formed the basis of the workload model. This comparison emphasizes the similarities as

[1] PPTV home page, http://www.pptv.com/

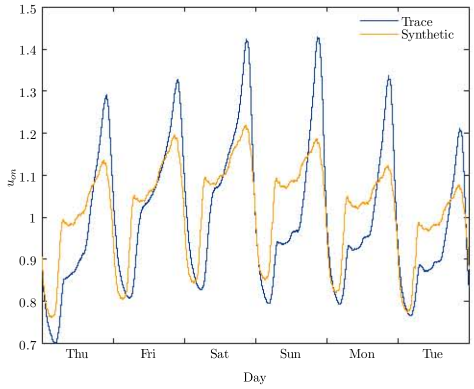


Figure 9.1: Comparing the fraction of online users in the synthetic workload with the trace data.

well as the approximation errors. In the latter situation, we compare our synthetic workload with data from other studies of IPTV systems, with the intention of showing that the workload may characterize a broader pattern on how users watch TV.

9.2 Evaluation Against the Trace Data

We begin by comparing the synthetic workload with the IPTV data available from Qiu et al., which is the basis behind our extended model and synthesis algorithms. To this end, we compare three essential metrics: the fraction of online user equipments, the session characteristics, including the session rate and session duration, and, finally, the channel popularity. We indicated previously that we do not model and synthesize the user classes, and user preference for particular TV channels, due to insufficient data covering these aspects.

9.2.1 User Activity

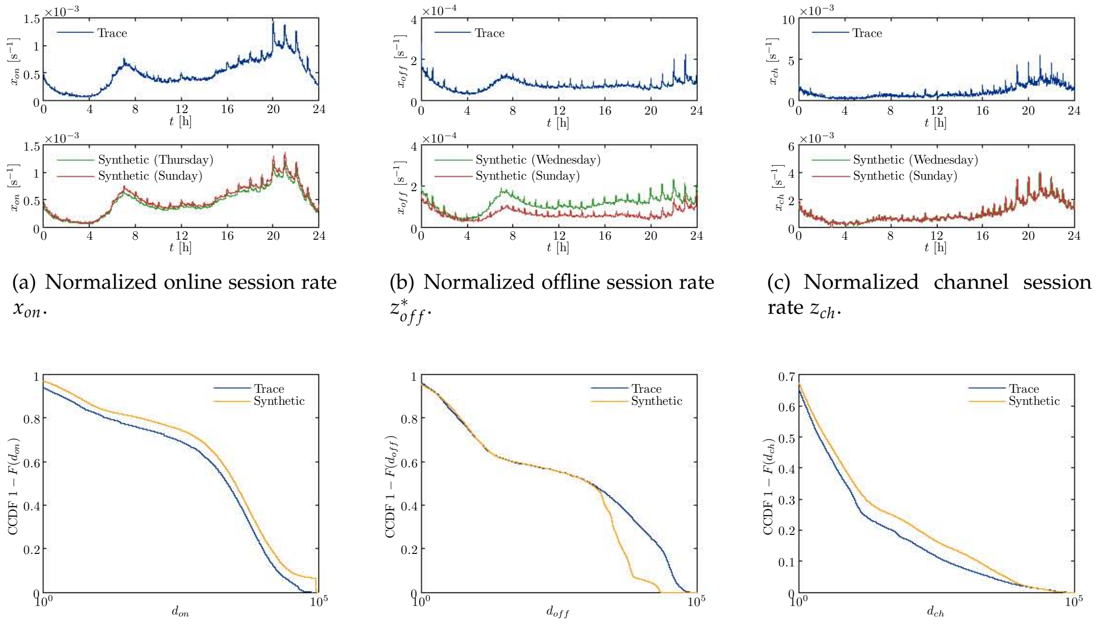
9.2.1.1 The Fraction of Online Users

The figure 9.1 compares the fraction of online UEs, u_{on}, between the initial trace data, and the synthetic workload. We remind the reader that our algorithms, do not generate u_{on} directly. Instead, the fraction of online/offline users are determined automatically by the rate of the online/offline session events.

In assessing the goodness of our results, we note that the trend of both the trace and synthetic quantities are similar throughout the measurement time window. First, the online UE fraction during a day follows the pattern of low activity during the night, and a high activity during the afternoon and evening hours. Second, the addition of the modulating function approximates well the oscillatory weekly pattern, with more users watching TV during the weekend.

Despite this positive outcome, there are several observable errors, where the synthetic workload underestimates the user demand during peak hours and only some days offer a good match. The root cause of these errors lies in the nature of the session rate model, which, in turn, determines the fraction of online users. Due to the limited data that was available, the session rate model uses only on day worth of data. As representative as the day may be, it is impossible to capture the changes between days. Therefore, the workload represents the fraction of online users more accurately during Friday and Saturday than Thursday or Sunday.

In addition, the underestimation of the evening peaks is determined by the removal of the high transient components from the session rate. In order to keep the model parameters low, our model excludes the very high frequency components, the loss of which causes a lower arrival rate at the



(a) Normalized online session rate x_{on}.

(b) Normalized offline session rate z^*_{off}.

(c) Normalized channel session rate z_{ch}.

(d) Online session length d_{on}.

(e) Offline session length d_{off}.

(f) Channel session length d_{ch}.

Figure 9.2: Comparing our synthetic workload with the trace data.

TV program boundaries in the evening. The same effect, albeit less pronounced, is observed after midnight when the users turn off their equipment, and the synthetic departure rate is also smaller.

Unfortunately, given the complexity of real-life data, correcting these errors is not easy, and for our purposes, despite the existing imperfections, we regard the existing model as acceptable. For other applications, where a higher degree of precision is desirable, the session rate may simply be described by a higher number of parameters, by using the same principles we described in the chapter 7.

9.2.1.2 The Session Rate

The figures 9.2(a), 9.2(b), and 9.2(c) compare the trace and the synthetic online, offline and channel session rate with a model based on 58, 47 and 64 frequency components, respectively. Like in the original data, the online session rate is normalized to the number of offline users, while the offline and channel session rates are normalized to the number of online users. Because in the synthesis algorithms the session rate takes precedence over the other workload characteristics, we obtain a very good approximation of the real data.

The online and offline session rates are modulated to generate the weekly pattern for the fraction of online users. To this end, our figures show the synthetic rate for two representative days of the week, Thursday and Sunday, when the user demand is low and high, respectively. The rate variation is similar for both online and offline events, with the online rate slightly higher and offline rate lower during peak days. In our mode, the channel session rate is not affected by the day of the week.

9.2.1.3 The Session Duration

In the chapter 8, our analytical analysis showed that, due to incomplete data and model approximations, there exists a mismatch between the number of session events and the session intervals, generated at the model rate and duration distribution. This mismatch occurs for any session type. In order to accommodate this difference, there are two possibilities: modifying the workload

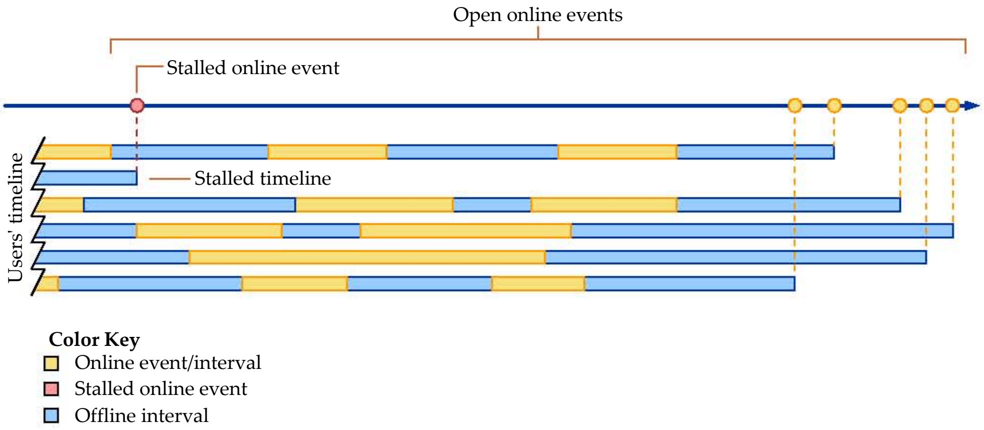


Figure 9.3: Stalled user timeline during the workload synthesis.

model or prioritizing one parameter over the other during the synthesis. Because both methods introduce a deviation from the available data, we examined and implemented the second approach, by fixing the generated rate to the one specified by the model, and allowing a certain in error for the interval lengths. The method has the advantage of keeping the workload model simpler, while providing a practical solution for the difference.

The online and the offline sessions belong to the same user timelines, and in theory the error is shared between them. Previously, we calculated the ratio between the number of sessions, fitting into an arbitrary time period, based on the interval length, d, and absolute event rate, z, at:

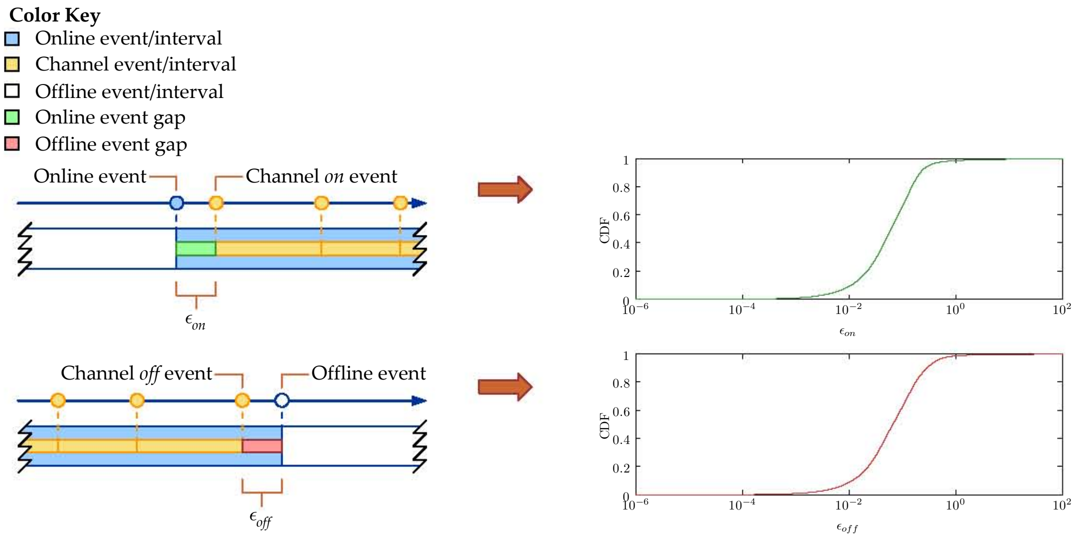
$$\frac{n_{d,o}}{n_{z,o}} \approxeq 0.65. \tag{9.1}$$

This result indicates that the online plus offline interval lengths are larger and do not fit within the session events. The figures 9.2(d) and 9.2(e) reflect the same difference in the complementary cumulative distribution functions (CCDF) of the online and offline intervals, respectively. We note that the error is shared unequally between online and offline sessions, because of the way our implementation of the synthesis algorithms handle the event assignments from the open sets. In fact, the average length of the online intervals is slightly higher than the average, which is compensated by a larger decrease in the average of offline sessions.

In the interest of keeping the description of the synthesis algorithms as simple as possible, in the chapter 8, we omitted several subtleties. One of them is the prevention of *stalled user timelines*, which are user timelines that do not progress because their event is never selected from the open set. The figure 9.3 illustrate the situation of a stalled timeline, when the corresponding online event lags behind the normal progress. As the distance to the current event continues to increase, the probability of ever selecting the stalled event approaches zero.

In our C++ code, we tested a number of strategies to prevent the occurrence of stalled timelines, each with a different level of performance in terms of execution time, approximation of the model, and balance between online and offline sessions. We believe that our selected approach offers a reasonable compromise between these, with the side effect of longer than normal online intervals, in particular because the design of the session coordination and peer coordination algorithms is not very sensitive to the session length. We acknowledge that more refined and smarter synthesis algorithms are possible, which can reduce the error between the generated workload and the model.

In the figure 9.2(f), we examine the interval duration for channel sessions. The error is of similar nature, due to the model mismatches and algorithm constraints. The previous analysis shows that the number of intervals accommodated by an arbitrary period, based on duration,



(a) The cause of the event error in the session timeline.

(b) The CDF of the event error.

Figure 9.4: Approximation error between online/offline sessions and channel sessions.

exceeds the one based on the channel event rate:

$$\frac{n_{d,c}}{n_{z,c}} \approxeq 1.39. \tag{9.2}$$

Hence, the mean synthetic interval is slightly higher than the model, which is also reflected in our result.

9.2.1.4 The Session Event Gap

In the model data available from Qiu et al., the channel events represent any state change for the current TV channel of a user, including switching on and switching off the user equipment (Qiu, Ge, Lee, Wang, Xu, & Zhao, 2009). When the user turns on its UE, it automatically tunes into, usually the last, TV channel; and, more obvious, when the user turns off its UE, it also tunes out of current channel. By considering this logic flow, in the timeline of each user we divided the channel events into *on*, *off* and *change* flavors. From their definition, the first two types are always mapped to online and offline channel events, where ideally they should coincide.

In practice, considering the stochastic nature of the event rates, it is impossible to obtain a perfect overlap between online and *on* channel events, and offline and *off* channel events. Our synthesis algorithm takes this approximation into account, and the general rule is that the *on* channel event should immediately succeed the online event, while the *off* channel event should immediately precede the offline event. The events matching in this way should be as close as possible, with a negligible gap error denoted by ϵ_{on} and ϵ_{off} as we show in the figure 9.4(a).

To evaluate the event matching of our synthetic workload, in the figure 9.4, we show the CDF of the online and offline session event gap, that is the distance between a pair of matching events. The result shows that the matching is very good, with a session gap error median of 61.7 ms and 66.5 ms, respectively. The error is greater than one second in less than 1.5 % of sessions. The session gap error is very small, its origins are purely artificial and the effects have little impact on the overall system performance. Therefore, we believe that eliminating the error completely, by modifying the workload model or the synthesis algorithm, would bring little benefit to worth the effort.

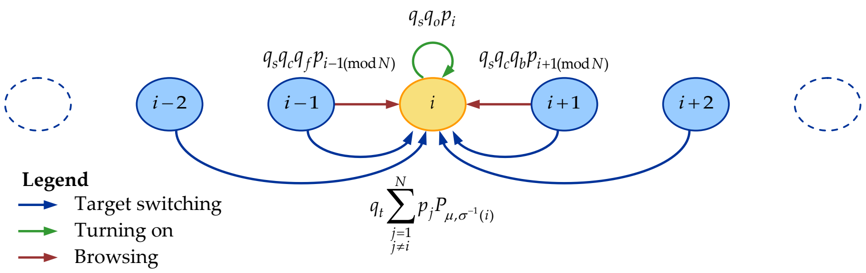

$q_s q_o p_i$
$q_s q_c q_f p_{i-1(\text{mod}\,N)}$
$q_s q_c q_b p_{i+1(\text{mod}\,N)}$
$i-2$
$i-1$
i
$i+1$
$i+2$
Legend
Target switching
Turning on
Browsing
$q_t \sum_{\substack{j=1\\ j\neq i}}^{N} p_j P_{\mu,\sigma^{-1}(i)}$

Figure 9.5: Modeling the channel switching with a Markov chain.

9.2.2 Channel Popularity

The channel popularity consists of several components that model different aspects of how the human user changes between TV channels. The TV viewing behavior is divided between tuning the TV set to a particular channel of interest, called *target switching*, and browsing between the available channels with the intention of finding an interesting program, called *sequential switching*. As we described in the section 7.3, Qiu et al. conclude that a set of uniform probabilities can model the different types of channel switching. For target switching, the channel access frequency follows a combination of zipf and exponential for most popular and the rest of the channels, respectively.

9.2.2.1 Global versus Target Popularity

In order to independently verify the workload channel popularity, we calculate the access probability of a channel, denoted by p_i, resulting from both target and sequential switching. For this purpose, we model the channel switching process using a time-independent Markov chain, where each state represents a TV channel (Bikfalvi et al., 2011).

The state transitions in the Markov chain depend on the channels order as accessed by the user. For this purpose, we use the random permutation function $\sigma(i)$, introduced in the section 7.3.3, that is unique for every user and maps the set of TV channels ordered by their decreasing target switching popularity $P_{\mu,i}$ to the order in which channels are stored at the user's TV set. In order to simplify the notations, we shall define the transition Markov chain in the channels order received by the user. Therefore, a TV channel on the position i in the Markov chain will have the target switching popularity $P_{\mu,\sigma^{-1}(i)}$. The figure 9.5 illustrates the Markov state transitions when tuning in to a given channel i:

- the transition probability of target switching from a channel j is $q_t P_{\mu,\sigma^{-1}(i)}$, where $j \neq i$;
- the turning-on transition probability is q_o;
- the transition probability of forward sequential switching from the preceding channel is $q_s q_c q_f$, and;
- the transition probability of backward sequential switching from the succeeding channel is $q_s q_c q_b$.

Hence, we can write the probability of switching to channel i (i.e. the transition probability to the Markov state i) as:

$$p_i = q_t \sum_{\substack{j=1\\ j\neq i}}^{N} p_j P_{\mu,\sigma^{-1}(i)} + q_s q_o p_i + q_s q_c q_f p_{i-1(\text{mod}N)} + q_s q_c q_b p_{i+1(\text{mod}N)}. \tag{9.3}$$

By considering the transition probabilities for all TV channels and the constraint that the sum of all states equals one, we obtain the nonhomogeneous linear system of $N+1$ equations with N unknowns:

$$\begin{cases} \beta p_1 + \gamma_1 p_2 + \alpha_1 p_3 + \alpha_1 p_4 + \cdots + \alpha_1 p_{N-1} + \delta_1 p_N & = 0 \\ \delta_2 p_1 + \beta p_2 + \gamma_2 p_3 + \alpha_2 p_4 + \cdots + \alpha_2 p_{N-1} + \alpha_2 p_N & = 0 \\ \alpha_3 p_1 + \delta_3 p_2 + \beta p_3 + \gamma_3 p_4 + \cdots + \alpha_3 p_{N-1} + \alpha_3 p_N & = 0 \\ \alpha_4 p_1 + \alpha_4 p_2 + \delta_4 p_3 + \beta p_4 + \cdots + \alpha_4 p_{N-1} + \alpha_4 p_N & = 0 \\ \vdots & \\ \gamma_N p_1 + \alpha_N p_2 + \alpha_N p_3 + \alpha_N p_4 + \cdots + \delta_N p_{N-1} + \beta p_N & = 0 \\ p_1 + p_2 + p_3 + p_4 + \cdots + p_{N-1} + p_N & = 1, \end{cases} \tag{9.4}$$

where, for convenience, we have made the temporary notations:

$$\begin{aligned} \alpha_k &= -q_t P_{\mu,\sigma^{-1}(k)}, \\ \beta &= 1 - q_s q_o, \\ \gamma_k &= -q_t P_{\mu,\sigma^{-1}(k)} - q_s q_o q_f, \\ \delta_k &= -q_t P_{\mu,\sigma^{-1}(k)} - q_s q_o q_b. \end{aligned} \tag{9.5}$$

Written in matrix form, we have:

$$\mathbf{A} \cdot \mathbf{P} = \mathbf{B}, \tag{9.6}$$

where $\mathbf{A}$ is the $(N+1) \times N$ matrix:

$$\mathbf{A} = \begin{bmatrix} \beta & \gamma_1 & \alpha_1 & \alpha_1 & \cdots & \alpha_1 & \delta_1 \\ \gamma_2 & \beta & \delta_2 & \alpha_2 & \cdots & \alpha_2 & \alpha_2 \\ \alpha_3 & \delta_3 & \beta & \gamma_3 & \cdots & \alpha_3 & \alpha_3 \\ \alpha_4 & \alpha_4 & \delta_4 & \beta & \cdots & \alpha_4 & \alpha_4 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \gamma_N & \alpha_N & \alpha_N & \alpha_N & \cdots & \delta_N & \beta \\ 1 & 1 & 1 & 1 & \cdots & 1 & 1 \end{bmatrix}, \tag{9.7}$$

$\mathbf{P}$ is the $N \times 1$ matrix:

$$\mathbf{P} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 & \cdots & p_N \end{bmatrix}^\top, \tag{9.8}$$

and $\mathbf{B}$ is the $(N+1) \times 1$ matrix:

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix}^\top. \tag{9.9}$$

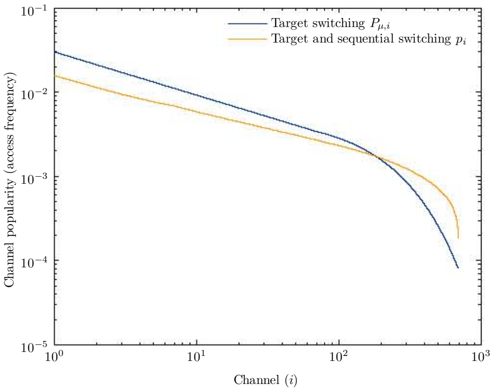


Figure 9.6: Comparing the target switching mean channel popularity, $P_{\mu,i}$, with the mean channel popularity for target and sequential switching, p_i.

Since the first N equations are linearly dependent[2], the system is equivalent to a non-homogeneous linear system of N equations with N unknowns. Given our parameters, the remaining N equations are linearly independent, for their determinant is different from zero, and the system has the unique solution:

$$\mathbf{P} = \mathbf{A}^{-1}\mathbf{B}. \tag{9.11}$$

Due to the sequential switching and random permutation of the channel order, the distribution of the total channel access probability p_i is different from the target switching channel popularity. The figure 9.6 compares the channel popularity, measured as access frequency, with and without sequential switching. The result shows that sequential channel switching redistributes a fraction of the popularity from the most popular channels to the least popular channels.

9.2.2.2 Target Switching Popularity

The work by Qiu et al. provides an extensive data set on the channel popularity. The authors approach the measurement problem from two perspectives: the definition of the channel popularity, and the temporal aspects of the measurement. The first recognizes that the popularity may be represented as both the relative *access frequency*, and the total amount of time the users spend, called *dwell time*, on the given channel. Both the measurement study and our own previous work on the issue (Bikfalvi et al., 2011) shows that, in practice, the access frequency and dwell time yield similar popularity values when the mean session duration or channel holding time are equal.

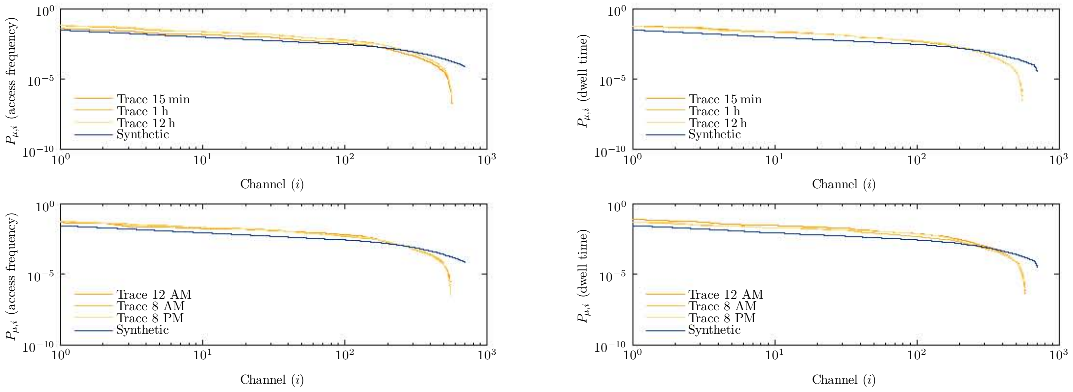
The temporal dimension of the popularity measurement concerns the time of day and the length of the observation window during which the measurement is taken. To this end, the authors of Simulwatch provide three data sets for each case: varying duration of 15 minutes, 1 hour and 12 hours; and varying time of day at 12 AM, 8 AM and 8 PM, respectively. Unfortunately, their data focuses exclusively on the target switching component of the mean popularity, whereas the sequential switching or browsing is characterized separately.

The figures 9.7(a) and 9.7(b) compare the synthetic channel popularity with the trace, where the mean channel popularity is measured as access frequency and dwell time, respectively. For the trace data, we include the temporal dimensions, with the popularity as function of the

[2] The first N equations are linearly dependent due to the constraint:

$$\sum_{i=1}^{N} P_{\mu,\sigma^{-1}(i)} = 1. \tag{9.10}$$

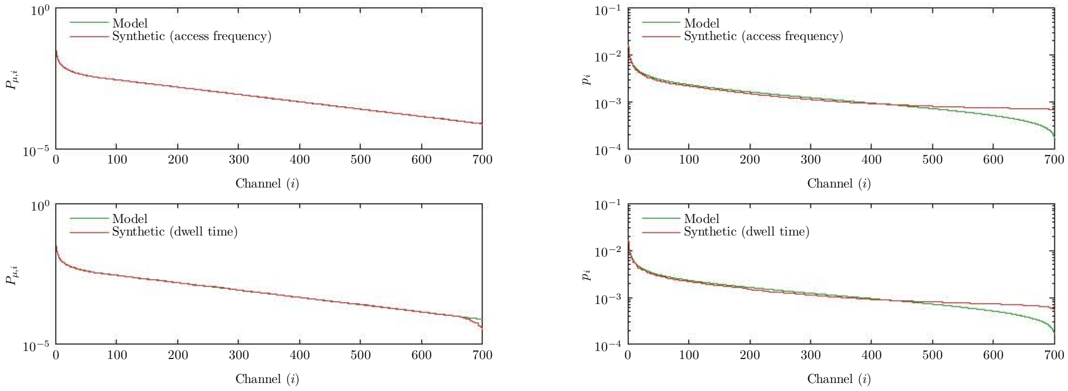
Therefore, any equation can be written as a linear combination of the other $N-1$ equations.



(a) The popularity measured as access frequency.

(b) The popularity measured as dwell time.

Figure 9.7: Mean channel popularity for target switching P_μ: trace versus synthetic workload.



(a) Mean popularity for target switching, P_μ.

(b) Mean popularity for target and sequential switching, p.

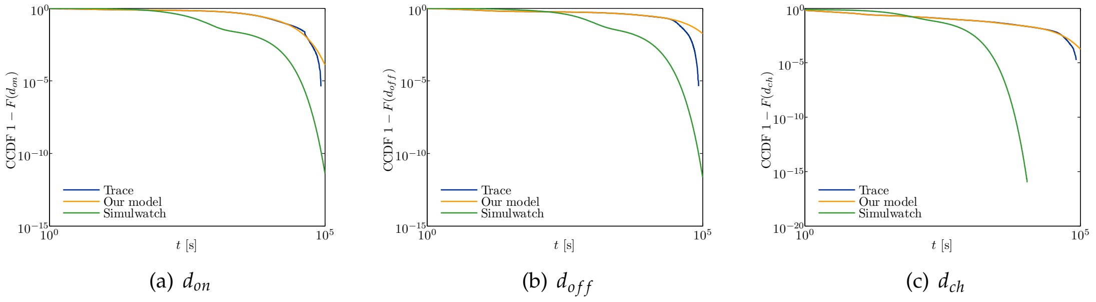
Figure 9.8: Mean channel popularity: model versus synthetic workload.

measurement time illustrated in the top figure, and the popularity as function of the time of day shown on the bottom.

Qiu el al. conclude that the temporal dimension has little effect on the overall popularity, where the channels are always sorted in the order of their decreasing popularity. The authors of the study leave open the question on whether the unsorted channel popularity changes with the size of the measurement window, or with the time of day. For example, a subset of TV channel is more popular in the morning hours than in the evening. Instead, their model assumes a uniform mean channel popularity, we denoted by P_μ, and accommodates the popularity dynamics with the addition of an instantaneous component P_t.

Unfortunately, because of the lack of any solid data, we assume that the mean popularity is the same throughout the day and for any measurement interval. By neglecting these imperfections of the model, we observe that the synthetic popularity approximates well the collected data for a large subset of the existing channels for both the access frequency and the dwell time. The largest error occurs for the most unpopular channels, where the popularity distribution decreases faster than the exponential model.

In the figures 9.8(a) and 9.8(b), we compare the target and global popularity with the corresponding theoretical values. This result shows that the synthesis algorithm approximates the model well, with the exception of a minor error for the dwell time of the most unpopular channels.



(a) d_{on} (b) d_{off} (c) d_{ch}

Figure 9.9: Comparing the distribution of the session length: trace data, our model and Simulwatch model.

Parameter	p_1	p_2	p_3	λ_1	λ_2	λ_3
d_{on}	0.3	0.66	0.04	$1.3 \cdot 10^{-2}$	$3.3 \cdot 10^{-3}$	$2.3 \cdot 10^{-4}$
d_{off}	0.19	0.75	0.06	$3.2 \cdot 10^{-2}$	$2.5 \cdot 10^{-3}$	$2.4 \cdot 10^{-4}$
d_{ch}	0.23	0.64	0.13	2.1	$2.6 \cdot 10^{-2}$	$3.2 \cdot 10^{-3}$

Table 9.1: The Simulwatch parameters for the hyper-exponential distribution modeling the session length.

The cause of this error is the limited observation period of the user activity (7 days), and the small number of users (10 000 users). Generally, we expect that the dwell time popularity converges to the access frequency after a sufficiently large period of time that guarantees the same mean channel session intervals.

Although the channel popularity model is not perfect, in our opinion, the approximation is acceptable for an adequate design and testing of the application server control algorithms. Our results show that the main error is generated by the model, and not by the synthesis algorithms. Therefore, in situations where a more accurate match is preferred, future work may extend the existing zipf-exponential model to address the distribution of the unpopular channels.

9.3 Comparison with the Simulwatch Workload Generator

Our work on the modeling and the synthesis of a large scale IPTV workload extends the previous efforts by Qiu et al., and their IPTV workload generator, called *Simulwatch*. Our improvements focus on increasing the accuracy and correcting several flaws in their model, and providing a more rigorous approach to the implementation. In this section, we aim to justify our work, by examining the differences between our and their contribution.

9.3.1 Session Length

Simulwatch uses a mixture of exponentials distribution to model the session length d_{on}, d_{off}, and d_{ch}. However, an analysis of the parameters indicates that the probability distribution is only a rough approximation of the trace data. The figure 9.9 compares the empirical distribution resulting from the actual data, with the distributions proposed by our model and Simulwatch model, respectively. One may observe, that while the hyper-exponential can be used to approximate the data, the parameters proposed by Simulwatch, given in table 9.1, do not guarantee an excellent goodness-of-fit.

9.3.2 Session Rate

Capturing the daily pattern of the session event rate is a challenging task. In particular, the temporary increase in the user activity at TV program boundaries, or flash-crowd, has a significant importance for a correct assessment of the system performance. Recognizing this, the authors of Simulwatch propose using a signal analysis technique that intends to model the complex properties of the session rate signal in the frequency domain. In their model, the bulk of the session rate spectrum (i.e. the Fourier transform) is approximated by the probability density function of a Weibull distribution, while the several spectral spikes at 1 hour, 30 min, etc. are modeled individually.

While their method shows promise and despite the results shown in the paper [3], the Simulwatch model does not work for the following reasons.

- By approximating the spectrum, we do not obtain an approximation of the signal. Even a small change in the frequency domain results in significant changes in time domain.
- The Fourier transform is a function on complex numbers. The modulus of the spectrum gives the magnitude of each spectral component, while the argument represents the phase. Both components are essential for a correct representation of the signal. However, Simulwatch does not.
- The PDF of the Weibull distribution tends to infinity at zero for a parameter $k < 1$. Hence, the model does not capture the $f = 0$ component.

The figures 9.10(a) and 9.10(b) illustrate why the Simulwatch model fails, in an example of the online session rate, x_{on}. We use the Simulwatch model of the frequency characteristic as described in (Qiu, Ge, Lee, Wang, Xu, & Zhao, 2009), where the power spectrum is described by the Weibull probability density function:

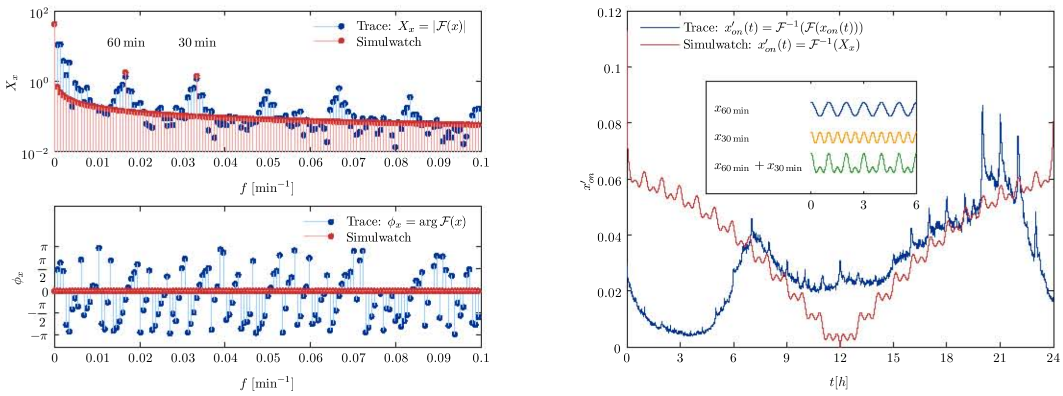
$$X_x^2(f) = \frac{k}{\mu}\left(\frac{f}{\mu}\right)^{k-1} \mathrm{e}^{-(f/\mu)^k}, \tag{9.12}$$

with $k = 0.0036$, $\mu = 278$. The authors model individually the spectral components corresponding to the spectral local maxima at 60 and 30 minutes: $X_x(1/60) = 1.76$ and $X_x(1/30) = 1.41$. As normal, the power and amplitude spectra are symmetric, centered around $f_s/2$, where f_s is the sampling frequency. The model does not propose a function for the phase, and we assume that it is zero.

Due to the aforementioned reasons, the reconstructed signal from the Simulwatch model, shown in the figure 9.10(b), presents the following properties. Because the phase is missing, the signal in time domain is a sum of cosine functions with zero phase and, hence, symmetric around zero. Next, the high frequency components are significantly attenuated with respect to the original signal. In fact, only the dominant components that were set manually at 60 and 30 minutes are easily distinguishable. We emphasize these signals in the smaller figure with their corresponding sum.

Because of these findings, we concluded that the Simulwatch model is inadequate for capturing the complex pattern of the session rate. Instead, our alternative based on dominant frequency components is much more suitable and yields a very close match to the original. Its disadvantage consists in a larger number of parameters that characterize the model, usually several tens of complex values, as opposed to the 6 or 8 parameters per signal proposed by Simulwatch. However, we believe that concerns related to a larger number of model parameters are purely academic, since they have little performance impact on the workload generation.

[3] In (Qiu, Ge, Lee, Wang, Xu, & Zhao, 2009), the figure 10 shows a near-perfect match between the real trace and the model fitting. Given that the Weibull approximation of the spectrum is in practice a low-pass filter, it is impossible to obtain the high-frequency components as that figure shows.



(a) The frequency analysis of the session rate (magnitude X_x and phase ϕ_x) and Simulwatch model.

(b) Reconstructed session rate in the time domain based on the spectral model.

Figure 9.10: The Simulwatch model for the session rate.

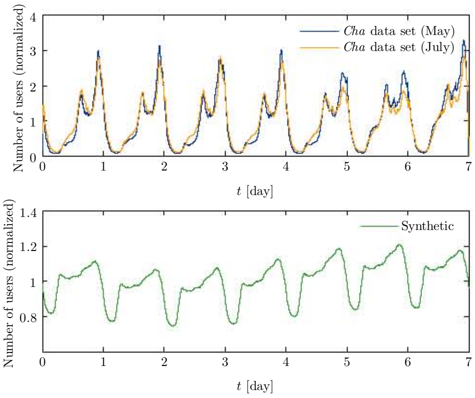


Figure 9.11: Comparison of the number of online users, normalized by the average.

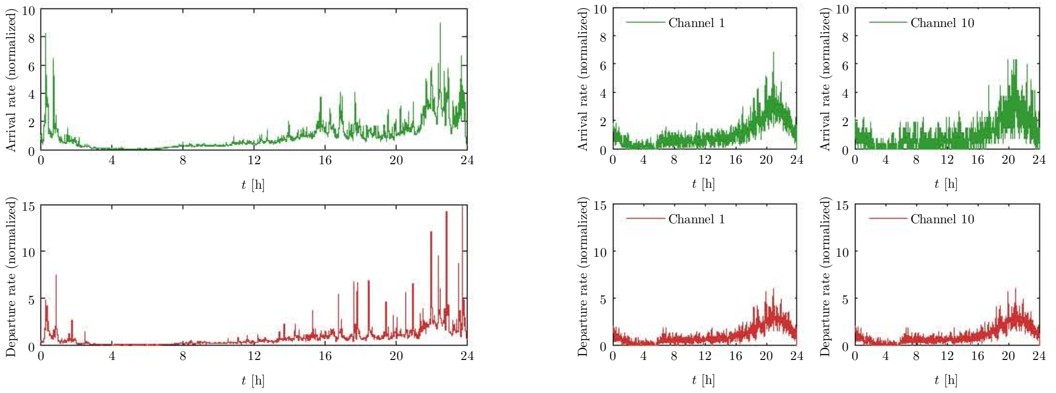
9.4 Comparison with Third-Party Data

Finally, we must answer the question: how representative is the IPTV user activity we described in the present work? The data published by Qiu et al. represents, without a doubt, an exhaustive set. It captures both the channel popularity of an IPTV provider with a rich channel package, as well as the activity of a large subscriber base spread across several time zones. In order to answer this question, we compare our synthetic workload with measurements from third-party IPTV systems.

Cha et al. published an extensive study on how users watch TV, relying on traces collected from a telco in Spain (Cha, Rodriguez, Crowcroft, et al., 2008). Because their work does not publish all variables from our workload, we examine only those that are available. We do not expect to obtain a perfect match, but rather to estimate the similarity between two different systems. To this end, we compare the following characteristics of the synthetic workload:

- the number of online users;
- the daily user arrival and departure rate on a selected TV channel, and;
- the distribution of the user inter-arrival and inter-departure time on a selected TV channel.

The figure 9.11 compares the number of online users, normalized by the average, between the *Cha* data set and our synthetic workload. The main characteristics of the data are the same, with



(a) *Cha* data-set: user arrival rate (top), user departure rate (bottom).

(b) Synthetic workload: channel 1 (left), channel 10 (right)

Figure 9.12: The user arrival and departure rate on/from a popular TV channel.

daily patterns consisting of a low number of viewers at night, slightly increasing in the morning, another subsequent decrease during the day, when most subscribers are at work, and a high activity during evening hours.

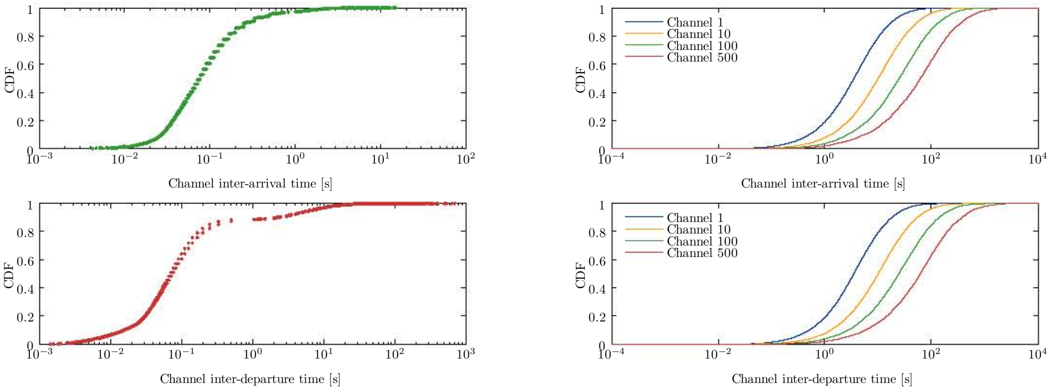
However, we also observe several key differences, which are most likely generated by the cultural differences between the United States and Spain. One of them, is the lower evening demand during weekend evening, such as Saturday and Sunday, thereby altering the weekly variation. Then, the absolute viewership during the night is near zero, unlike our synthetic workload where the fraction of online users is substantial (around 60–70 %). While it is impossible to assess the true cause of this difference, it may be cause by several factors, including: (i) the number of subscribers in Spain is much lower and most of them turn off their TV set at night; (ii) the equipment installed by the IPTV provider and whether it continues receiving the channel even when the TV set is turned off.

Next, we compare the channel traffic in terms of user arrival and departure rates, which are the online/offline session event rates. Cha et al. do not provide results on the global rates, but instead they focus on the most popular TV channel from the *free* TV package, available to all subscribers. Because the channel offer between the two data-sets is different, 700 channels, in the United States, versus 120 TV channels, in Spain, we selected two representative channels, 1 and 10, from our synthetic workload. Both these channels are popular, while accommodating different fractions of users.

The figures 9.12(a) and 9.12(b) compare the two situations, and we make the following observations. The Spanish data-set is fairly consistent to the number of online users, presented previously. To this end, the daily user activity experiences two peaks in the 2:00 PM–6:00 PM, and again 10:00 PM–1:00 AM time frames. This result is unlike the daily activity in the United States, where the differentiation at mid hours during the day is not this obvious. Although the data represents the user activity on a single TV channel, due to the channels high popularity and otherwise uniform user access over time, the synthetic arrival and departure rates have the same shapes to the global event rates. Hence, we may reasonably conclude that although the *Cha* data illustrates a single TV channel, the pattern is identical for the global user activity as well.

In the figures 9.13(a) and 9.13(b), we examine the distribution of the user inter-arrival and inter-departure times. As the case of the event rate, the *Cha* data set is limited to the a single TV channel, and, in the interest of thoroughness, we compare with four representative channels from the synthetic workload that are spread across the popularity spectrum: 1, 10, 100 and 500.

In our opinion, these last results emphasize the similarities between the user behavior from



(a) *Cha* data-set

(b) Synthetic workload

Figure 9.13: The CDF of the user inter-arrival and inter-departure times on/from a popular TV channel.

two regionally different IPTV services. The data match is not perfect, because of the different settings of the systems, such as the available TV channels, or the user behavior, including cultural aspects. Data available from smaller scale IPTV deployments shows equally comparable patterns in the user participation and channel access (Yu et al., 2009). Although analyzing in further depth the user behavior is out of the scope of the present work, the positive examples that we examined in this section serve to validate the confidence in our synthetic model.

9.5 Conclusions

A good design of the session and peer coordination algorithms at the P2PTV-AS requires an accurate representation of the IPTV subscriber activity. Because our synthetic workload is reversed engineered from the existing and limited measurement data, it performs a number of approximations in generating the user activity process. These are generated by both the workload model, which is a simplified view of the reality, and the synthesis algorithms that are designed to be efficient and focused on a small set of parameters.

In order to verify whether these errors do not alter significantly the known data, in this chapter, we conducted an extensive evaluation of the synthetic workload. Our strategy includes comparisons against the original trace data, published by Qiu et al., with an emphasis on the following workload characteristics: the fraction of online users, the session rate, the session duration and the channel popularity. The results show that our workload offers a good approximation of the data, in the context of our design requirements. For applications requiring grater precision, we offered several suggestions for future improvement, such as extensions to the model and changes to the synthesis algorithm.

In addition to the aforementioned characteristics, we tested the synthetic workload against the Simulwatch generator and third party IPTV data. These results justify the present work, and show that although the current model is based on a single IPTV deployment, its properties can be generalized to a wider subscriber audience.

Part IV

The P2PTV Application Server

Chapter 10

Application Server Functions

10.1 Overview

The application server, or the P2PTV-AS, is the central component of the P2PTV service for the IMS. Like any SIP application server, it controls the viewers access to the television service, by responding to the session requests initiated by the user equipments. When the requested stream is uploaded by another UE peer, the AS intermediates the session establishment as a SIP back-to-back user agent, making the peer-to-peer connections entirely transparent for the users.

The IMS service provisioning and the signaling protocol is a matter of SIP and IMS standardization, and we have presented them extensively in the chapters 4 and 5. However, the most novel feature of the P2PTV-AS is the handling of the peer-to-peer participation. Like other real-time communication or entertainment applications, a fast response to users action is of the essence for the service to be perceived as a high-quality one. Unfortunately, while the P2PTV streaming benefits greatly from the IMS QoS reservation and commitment, the corresponding signaling procedure is a time-expensive operation. Unlike using IP multicast or a single broadcast server, uploading peers must establish or release streaming sessions every time the downstream UEs change their current TV channel.

In order to reconcile both the IMS signaling requirements and a low latency during peer-to-peer changes to the uploading multimedia sessions, we proposed a proactive approach where the UE peers have reserved sufficient network resources in the form of inactive upload sessions. The P2PTV-AS uses a *session coordination algorithm*, or SCA, to compute regularly the number of sessions that accommodate well the existing viewer demand for every TV channel.

To minimize the impact of churn, the UE peers only upload the TV channel streams, called *secondary streams*, that can be maintained after the viewer changes to a different channel. Generally, these are different from the *primary streams* of the current TV channel, and are downloaded until needed. Implicitly a peer may have inactive session only on the secondary streams. The P2PTV-AS uses a *peer coordination algorithm*, or PCA, to manage the allocation of the secondary stream, under the constraints of the available peer bandwidth, the user demand and the required number of

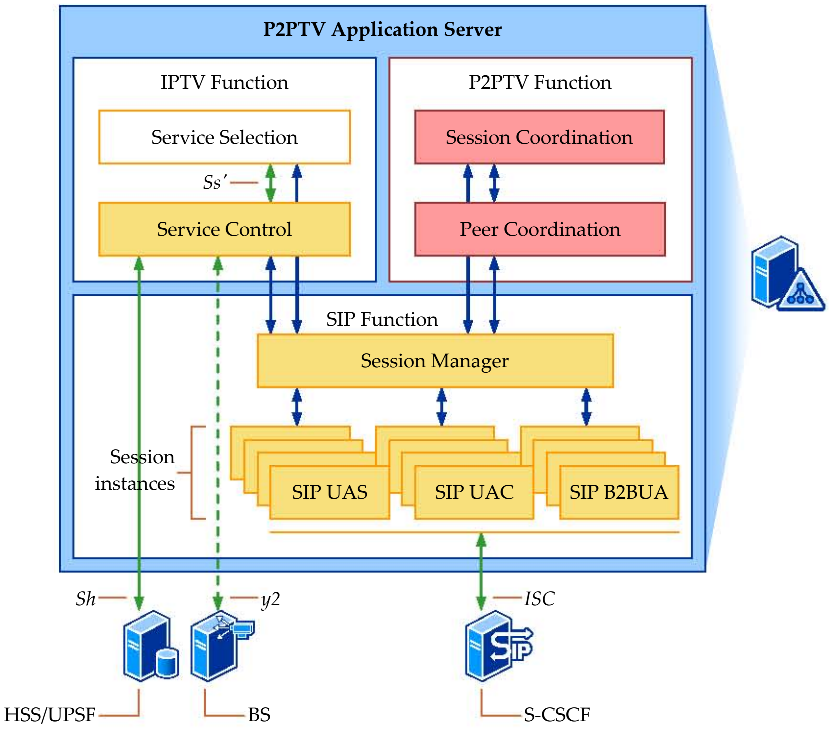


Figure 10.1: The main functions of the P2PTV application server.

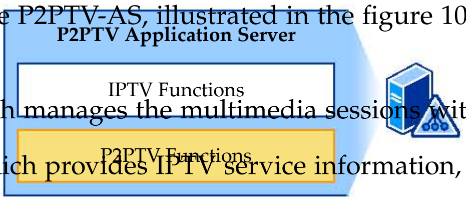
inactive upload sessions for each TV channel.

In the current and the following four chapters, we address the design of the P2PTV application server, with a special emphasis on the coordination algorithms, the SCA and the PCA. These form the core of the peer-to-peer infrastructure and are instrumental to the performance of the P2PTV service.

10.2 Architecture

The core functionality of the P2PTV-AS, illustrated in the figure 10.1, is divided in three components:

- the *SIP* function, which manages the multimedia sessions with the user equipments;
- the *IPTV* function, which provides IPTV service information, and;
- the *P2PTV* function, which coordinates the peer participation through the SCA and the PCA.



In the following, we shall describe each of these three components with a focus on the P2PTV function, as the essential innovation component of our P2PTV proposal.

10.2.1 The SIP Function

The architecture of the P2PTV-AS is based on the TISPAN specifications, where the IPTV functionality is divided between the service selection function (SSF) and the service control function (SCF) (TISPAN, 2011a). In TISPAN, there exists a third functional entity, the service discovery function, or SDF, but its implementation is out of the scope of our work. To this end, we assume that all user equipments know the public service identity (PSI) of the IPTV service (e.g. `sip:p2ptv@example.net`), and the service profiles in the subscriber database, HSS or UPSF, are configured with the an appropriate filter criteria.

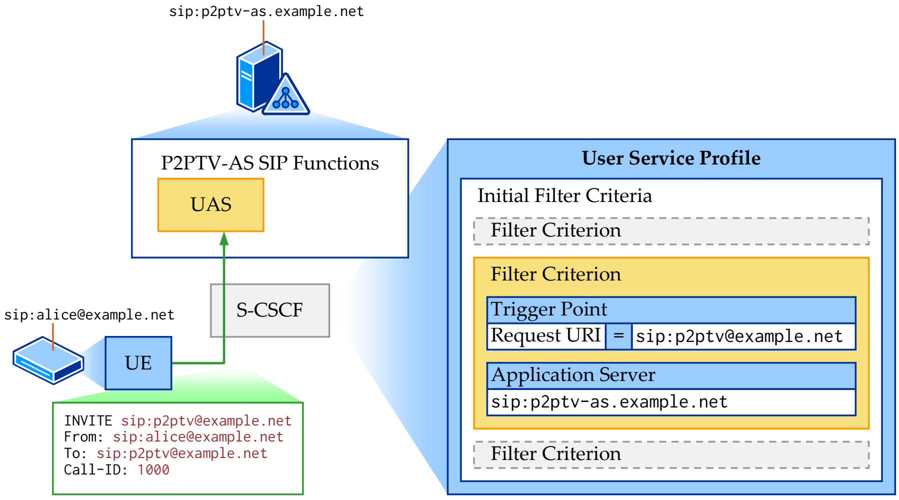


Figure 10.2: The P2PTV-AS acting as a server user agent (UAS).

The initial filter criteria (iFC) are a list of conditions applied to the SIP requests, called triggers[1], which, when are met, indicate to the S-CSCF to forward the SIP messages to a specified application server. The filter criteria are downloaded by the S-CSCF during the UE registration, and allow the S-CSCF to provide a specific service for every session. Multiple criteria, in the same user profile, are ordered by a priority number, and each criterion may contain several triggers[2]. In the appendix B, we describe detail the configuration of the subscription and service profile for the P2PTV service, in a testbed deployment using the FOKUS OpenIMSCore software. The configuration includes the definition of the filter criteria and service point triggers.

The P2PTV-AS uses the IMS Service Control interface, or *ISC*, to receive and send SIP session requests from or to the user equipments, via the S-CSCF. Depending on the type of handling at the P2PTV-AS and the nature of the required streaming or resource reservation, the multimedia sessions are classified in the following types:

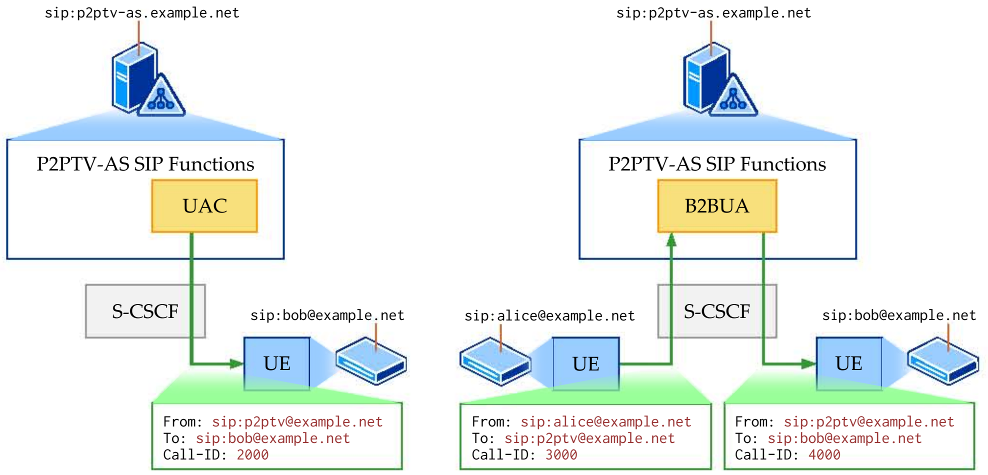
- *terminating sessions*, which originate at a UE during a request for a stream that is uploaded by the broadcast server;
- *originating sessions*, which are initiated by the AS with a foster peer for the purpose of creating an inactive upload session, and;
- *back-to-back sessions*, which are a pair of sessions, one terminating, one originating, used for peer-to-peer streaming.

The session type is an interchangeable property, meaning that a terminating and an originating session may be merged into a back-to-back pair, or, alternatively, a back-to-back pair may be divided in two separate terminating and originating sessions. The purpose of the session type is to allow the P2PTV-AS to assign the appropriate SIP user agent, as specified by the specification (Rosenberg et al., 2002): a user agent server (UAS) for terminating sessions, a user agent client (UAC) for originating sessions, and a back-to-back user agent (B2BUA) for back-to-back session pairs.

The figure 10.2 illustrates a terminating SIP session, serviced by a UAS instance at the P2PTV-AS. Terminating sessions are initiated by the user equipment, and they correspond to all download

[1] The complete name defined by the standard is service point trigger, or SPT (3GPP, 2012b).

[2] The name of initial filter criteria has historical reasons, where the IMS standards distinguish between initial and subsequent filter criteria (sFC). The iFC would apply to the initial requests, while the sFC are processed for subsequent requests within a SIP dialog. Because sFC conflict with the SIP routing rules for proxies, only iFC have been implemented (Camarillo & Garcia-Martin, 2011).



(a) The P2PTV-AS acting as a client user agent (UAC).

(b) The P2PTV-AS acting as a back-to-back user agent (B2BUA).

Figure 10.3: Cases of the P2PTV-AS SIP user agent.

sessions and the UE subscription to the P2PTV service information[3]. The figure emphasizes the filter criterion for the P2PTV service, which compares at the S-CSCF the equality between request URI of SIP messages and the P2PTV public service identity.

An uploading session originating at the P2PTV-AS uses a UAC instance. The P2PTV function coordinate the logic for initiating new upload sessions, either as a response to an outstanding UE request, or as an inactive dialog on behalf of the session coordination algorithm[4]. The figure 10.3(a) shows a SIP dialog, originating at the P2PTV-AS UAC. The P2PTV-AS is a trusted entity in the IMS core network, and therefore the routing of the originating sessions to the destination does not require the registration of the PSI.

To support the peer-to-peer streaming, a pair of a UAS and a UAC instances are joined as a single B2BUA instance. The user equipments maintain independent dialogs with the P2PTV-AS, and the P2P communication is transparent for them. The figure 10.3(b) illustrates a pair of sessions handled by the SIP B2BUA, where each dialog has a unique call ID. The B2BUA uses the P2PTV service logic, to map incoming and outgoing SIP messages on both the terminating and the originating dialogs, depending on the state of each sessions. In addition, the AS records the media information of each session, such that it may reply directly to a request, without involving the corresponding UE[5].

Exceptions, such as request cancelations or error responses, are handled by the P2PTV-AS B2BUA, whenever possible.

- If the exception occurs on the originating or uploading session, the P2PTV-AS will silently recover by either selecting a different upload UE peer, or the broadcast server.

- If the exception occurs on the terminating or downloading session, the P2PTV-AS will silently cancel the outstanding requests on the originating side.

[3] The appendix A, section A.2, summarizes all signaling procedures initiated by the UE.

[4] The appendix A, section A.3, summarizes all signaling procedures initiated by the P2PTV-AS.

[5] The session establishment when the uploading party is a foster peer, presented in the appendix section A.2.1.2 is a representative example of this situation, where the P2PTV-AS responds to the initial `INVITE` request, on behalf of the uploading UE.

10.2.2 The IPTV Function

The *service selection function*, or SSF, implements select requirements of the TISPAN IPTV standard (TISPAN, 2011a). For the purpose of the P2PTV service, the SSF maintains the list of available TV channels corresponding to each service package, for example basic, news, sport, kids, movies, etcetera. During the initial service subscription, each UE receives only the list of channels available to the profile of their current subscription. The description of every TV channel includes the characteristics of the associated audio/video streams.

The *service control function*, or SCF, manages the communication with the appropriate broadcast server for non-P2P sessions, communicates and accesses the user service profiles in the HSS or UPSF, validates the user access to each TV channel, and generates service notifications to the UE when the existing configuration changes. Although out of the scope of our present work, the SCF may implement any additional features supported by the standard, such as charging or tracking the user credit for pay-per-view content, program recommendation, or advertisement services[6].

10.2.3 The P2PTV Function

The objective of the P2PTV-AS design is to enhance the usage of the available UE peers and their bandwidth toward maximizing the service quality perceived by the user. Since multimedia sessions, once established, have reserved QoS resources within the IMS infrastructure, the remaining challenge is to accommodate the changes in the existing sessions that occur during TV channel switching.

In order to mitigate the signaling latency required for resource reservation, when establishing a new session to a TV channel, the P2PTV-AS searches for one or more foster UE peers that can upload the corresponding channel streams *and* already have established an inactive upload session with committed upload resources. If such peers are found, they are used to upload the stream to the new client. To accomplish this goal, in the chapter 4, we divided the problem in two main components. The first, called *session coordination*, determines the necessary number of upload sessions, for each TV channel, that accommodates the near future demand. The second, named *peer coordination*, computes the allocation of these sessions across the population of UE peers, given the necessity of TV channel and the constraint of UE network resources.

Unfortunately, as the available experimental measurements and models indicate, including our own, watching television is a highly complex activity that cannot be easily predicted. In the part III, we showed that even our simplified data analysis contains dynamic components that are hard to model, and we can safely assume that the workload diversity in real-life is even greater. For these reasons, we conclude that the design of our P2PTV-AS cannot focus exclusively on a set of static parameters such as our model, regardless of how accurate it may be. Instead, the P2PTV-AS software must adapt to the changing conditions of TV demand and adjust the allocation of peer resources. Toward this end, the remainder of this part details in greater depth the design of the P2PTV-AS session and peer coordination algorithms, the SCA and PCA.

10.3 The Session Coordination

10.3.1 Queueing Model

From the perspective of the session coordination algorithm, each TV channel i is similar to a queueing system, where new users arrive on the TV channel at the rate $z^*_{ch}|_i$, receive service for the duration of their channel session, $d_{ch}|_i$, and then leave the system or switch to a different channel.

[6] Because the focus of this work is broadcast television, we do not discuss SCF features such as video-on-demand, user-generated-content, or personal video recording

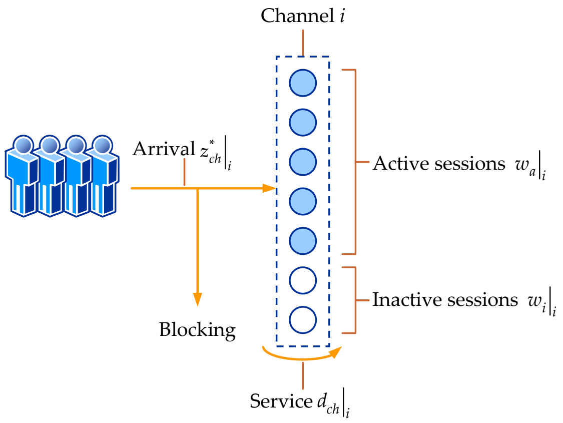

Channel i
Arrival $z^*_{ch}\big|_i$
Active sessions $w_a\big|_i$
Inactive sessions $w_i\big|_i$
Blocking
Service $d_{ch}\big|_i$

Figure 10.4: The queueing model of the system for the session coordination algorithm.

Each queue has multiple *servers*, corresponding to the number of upload session that are currently established. Some *servers* are busy, meaning they are active uploading session that may not be used by a different peer at the same time. Other *servers* are free, that is they are inactive sessions that can be assigned to new arrivals.

All channel queues have a zero-sized buffer: there is no waiting in the queue. A new channel viewer will be serviced by an inactive session right away, or it will resort to a different non-preferred mechanism to connect to the channel, such as establishing a complete end-to-end session to another peer, or using the broadcast server. Because the number of servers a.k.a. upload sessions is limited, some viewers may be *blocked* from connecting to an existing upload session. The *blocking ratio*, denoted by b, that is the fraction of arriving channel viewers that do not find a free session, reflects the P2PTV-AS performance of providing a good service.

On the other hand, the session *utilization*, denoted by ρ, that is the fraction of inactive upload sessions, indicates whether the network resources are locked into too many unused sessions. Ideally, we prefer having a very small blocking ratio, typically around a value set by the service provider, and a high utilization. The figure 10.4 illustrates the queueing model of a TV channel, where the blocking ratio and the utilization measure the system performance. The solution is the number of upload sessions for the channel i:

$$w = w_a + w_i\big|_i\,. \tag{10.1}$$

10.3.2 Static Solution

Using the Kendall's notation, the TV channel system is a G/G/w/w queue, where the arrival rate and the service time are described by a generic probability distribution. In this context, finding the number of upload sessions w, given the real-time characteristics of the user activity, is a challenging task. The first possibility is applying the queueing theory to the characteristics of our system, and determining w for a desirable reference blocking probability, denoted by $r_b = \beta$, and reference utilization, denoted by r_ρ, as illustrated in the figure 10.5. We consider the user activity, represented by a number of workload variables, such as the number of online users, u, the arrival rate to the TV channel, z^*_{ch}, etcetera, as an external disturbance that the SCA must compensate.

A *static solution* looks for a time-independent relationship between the disturbance, the input and the output of our system. For example, in a system with exponentially distributed and independent inter-arrival and service time, M/M/w/w, the Erlang B formula determines the blocking probability as a function of the user arrival and service rates, and the number of upload sessions:

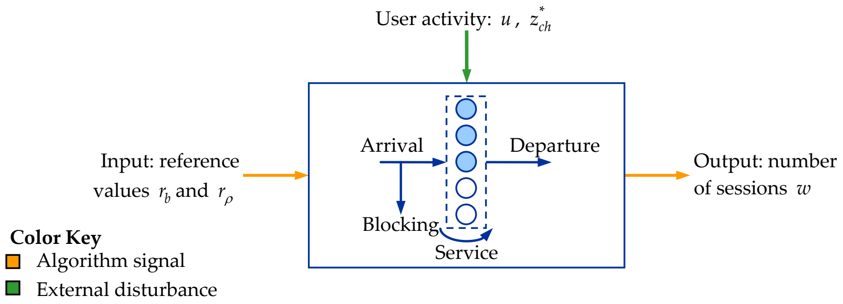
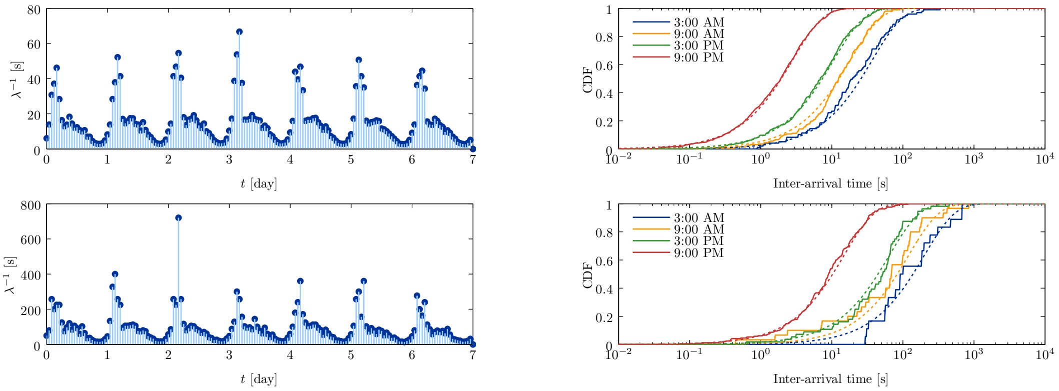

User activity: u, z^*_{ch}
Input: reference values r_b and r_ρ
Arrival
Departure
Blocking
Service
Output: number of sessions w
Color Key
🟧 Algorithm signal
🟩 External disturbance

Figure 10.5: The system diagram of the session coordination algorithm.



(a) The mean inter-arrival time.

(b) The CDF of the inter-arrival time: measured (solid line) and exponential distribution (dotted line).

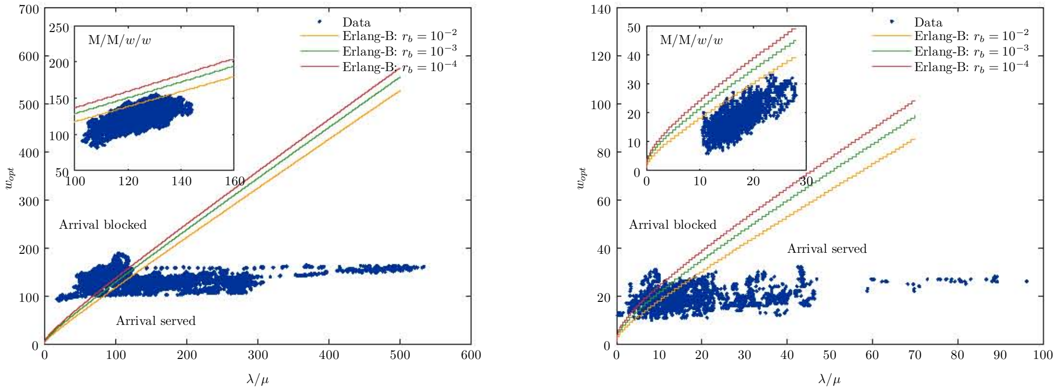
Figure 10.6: The user inter-arrival time on a popular (top) and unpopular (bottom) TV channel.

$$b = \frac{\dfrac{(\lambda/\mu)^w}{w!}}{\displaystyle\sum_{j=1}^{N} \frac{(\lambda/\mu)^j}{j!}}, \tag{10.2}$$

where b is the blocking probability (or ratio), w is the number of upload sessions, $\lambda = \bar{z}^*_{ch}|_i$ is the mean arrival rate and $\mu = 1/\bar{d}_{ch}|_i$ is the reciprocal of the mean duration on the selected TV channel.

Unfortunately, this approach presents a number of significant disadvantages. First, user arrival on the TV channel follows a non-homogenous distribution, where the mean number of channel arrivals per time unit is smaller in the mornings versus evenings, or in the middle of the weeks versus weekends. To this end, characterizing the user arrival is difficult, and it depends on the time scale used for observations. In the figures 10.6(a) and 10.6(b), we illustrate the user inter-arrival time on two distinct TV channels, one popular (channel 1) and one unpopular (channel 100). The measurement window is one hour, which represents a good compromise between a large interval, encompassing a statistically significant number of arrivals, and a small interval where the daily pattern is not dominant.

At a first sight and over small time scales, the inter-arrival times may be approximated by an exponential distribution characterized by the mean arrival rate. The figure 10.6(b) emphasizes this approximation, by comparing the CDF of the actual user activity (in solid line), with the theoretical probability distribution obtained from the sample mean. Because the mean changes with the daily time, we capture several snapshots, uniformly spaced across midnight, morning,



(a) Popular TV channel (channel 1).

(b) Unpopular TV channel (channel 100).

Figure 10.7: The optimal number of upload sessions versus λ/μ.

afternoon and evening hours.

This result is nevertheless misleading. Although we are tempted to consider the channel arrivals a Markov process, where the inter-arrivals are exponentially distributed, we must remember that the user activity is correlated to the TV program schedule, and therefore the arrival events are not truly independent. This situation becomes obvious with the next result, where we measure the relationship between system input and output. In the figures 10.7(a) and 10.7(b), we plot the optimal number of upload sessions, w_{opt}, versus the arrival to service rate ratio, λ/μ. The latter is an important indicator of the occupancy of the queueing system, by comparing the rate at which users arrive on a TV channel with the duration of their sessions. The optimal number of upload sessions is the value of w for which $b = 0$ and $\rho = 1$.

This result shows that there is no strong correlation between the optimal number of upload sessions and the channel arrival and service rates. In general, w_{opt} is bounded to a narrow interval around the mean number of channel users, regardless of λ/μ, for both popular and unpopular TV channels. This behavior is unlike an equivalent M/M/w/w system[7], with exponential independent inter-arrival and service times, shown in the smaller axes, where the Erlang-B equation determines a good upper boundary for the number of upload sessions, depending on the desired blocking probability, $r_b = \beta$.

10.3.3 Dynamic Solution using a Feedback Loop

Given the difficulties in exploiting the static characteristic of the channel queueing model to determine the number of upload sessions, we resort to the adaptive approach of a feedback control system. This *dynamic solution* does not require an exact law between a measurable property of the user activity (input) and the number of upload sessions (output). Instead, the rationale is to adjust the number of sessions such that the variables describing the performance of the queueing model, blocking ratio and utilization, are kept around the desired values.

Using a feedback control algorithm for the performance optimization of a computing system is a common approach (Hellerstein, Diao, Parekh, & Tilbury, 2004). To this end, our objective is to design a control system that considers the unique characteristics of the IPTV workload and the requirements of the P2PTV-AS. The P2PTV-AS is a discrete-time system, where the session coordination algorithm executes and takes corrective action at discontinuous moments of time. Such actions are synchronous with a sampling period T, with the sampling moment k occurring at the time kT.

[7] The equivalent M/M/w/w system has the same mean arrival rate and service time.

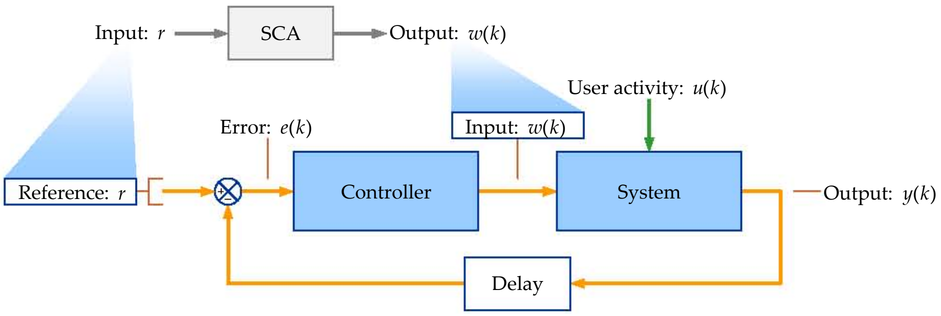


Figure 10.8: The principles of the feedback control at the session coordination algorithm.

Selecting the size of the sampling period considers two unique elements. The first is the desired reaction speed of the system to the changes in the user activity. The second is the delay required to establish an uploading session between the P2PTV-AS and the UE. In our implementation we choose a default sampling period of 10 seconds, which represents a reasonable compromise between the two criteria. At the end of the chapter 11, we discuss the implications of the sampling interval, and how its adjustment may influence the performance of the SCA.

10.3.3.1 Feedback Design Principles

The idea behind our design is the following. At every sampling moment k, the session coordination algorithm measures the blocking ratio, $b(k)$, and the utilization $\rho(k)$. These variables represent the *control output* of the system, generically denoted by $y(k)$. The *control error*, $e(k)$, between the measured output and the desired *reference* values, translates to the number of upload sessions, $w(k)$, called the *control input*. The figure 10.8 illustrates these principles. We note the distinction to the SCA system diagram, from the figure 10.5, where $w(k)$ represents the *output*.

The performance of a control system is characterized by the following properties:

- *stable*, if for a bounded control input signal, the control output is also bounded;
- *accurate*, if after a reasonable time the control error nears zero;
- *fast*, if the control output converges quickly to the reference value, and;
- *no overshoot*, if the convergence of the control output toward the reference is monotonic.

While all previous performance indicators are essential, the speed of convergence requires a special attention. Because the blocking ratio, $b(k)$, and the utilization, $\rho(k)$, are complementary quantities, in the interest of simplifying the design, we select the former as the control output[8]:

$$y(k) = b(k). \tag{10.3}$$

Unfortunately, the blocking ratio is a statistical property of whether the user arrivals receive service by an existing upload session. For this reason, it not possible to accurately estimate $b(k)$ within the duration of a single control sampling period. Especially on unpopular TV channels, simply there may not be any users arriving for many samples. On popular channels performing a

[8] In a more complex design, we could select both the blocking ratio and the utilization as a control output. For such a design choice, we obtain a multi-variable control system, where the output is a state vector of the form:

$$\mathbf{y}(k) = \begin{bmatrix} b(k) \\ \rho(k) \end{bmatrix}.$$

However, we are interested mainly in obtaining a small blocking ratio, even at the cost of smaller utilization. Because it is difficult to quantify the compromise between these metrics, in the present work the blocking ratio takes precedence.

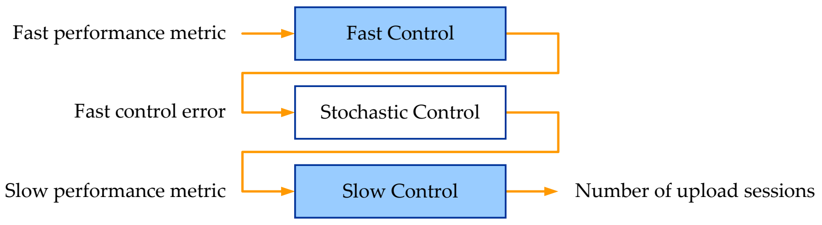


Figure 10.9: The overview of the control architecture for the session coordination algorithm.

blocking ratio reading on such a small interval may result in artificial fluctuations of the control output.

Traditionally, in telephony queueing systems, the blocking probability is determined based on the number of calls dropped during a longer duration such as 1 hour. We apply the same principle here. Given the selected sampling period of 10 seconds, we choose an averaging window for the blocking ratio, $n_w = 360$ samples. If we denote the total number of user arrivals, on the selected channel and during the sampling interval k, by $n_t(k)$, and the number of blocked arrivals by $n_b(k)$, then the blocking ratio is:

$$b(k) = \frac{\sum\limits_{i=k-n_w+1}^{k} n_b(i)}{\sum\limits_{i=k-n_w+1}^{k} n_t(i)}. \tag{10.4}$$

The averaging window is a smoothing filter on the blocking ratio during individual sampling intervals, and plays an essential role for the correct operation of the control algorithm[9]. The disadvantage of such a filter is the inherent delay of detecting the changes of the measured quantity, delay that is proportional to the window (or filter) size. Due to this constraint, it is impossible to rely on the blocking ratio as a fast indicator of the channel performance. Doing so would result in a control system that begins taking an appropriate corrective action long after the disturbance has occurred.

10.3.3.2 Control Architecture

To address these considerations, we propose a control architecture divided into the following components:

- a *fast control* loop, which uses a fast performance metric that is accurately determined during each sampling interval, and;
- a *slow control* loop, which uses the slower blocking ratio metric.

The objective of the first component is to ensure an immediate action to any disturbance resulting from the channel user activity, even though is does not drive the blocking ratio toward the reference value. The latter component measures the blocking ratio through the averaging filter, and introduces a corrective factor for the steady-state error.

Optionally, between the fast and the slow control loops, we may introduce a *stochastic control* function. Its purpose is to examine and compensate in real time any statistical properties of the fast error that are related to the blocking ratio $b(k)$. Although it is not a control system within the meaning of the feedback control theory, we maintain a similar name as it does reduce the

[9] The filter may be part of the measurement, or implemented in the control loop. However, the idea remains the same.

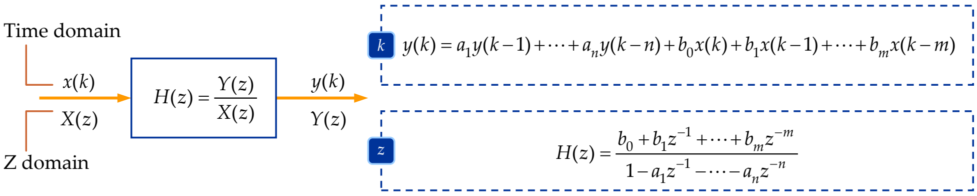

Time domain
$x(k)$
$X(z)$
Z domain
$H(z)=\frac{Y(z)}{X(z)}$
$y(k)$
$Y(z)$
k
$y(k)=a_1y(k-1)+\cdots+a_ny(k-n)+b_0x(k)+b_1x(k-1)+\cdots+b_mx(k-m)$
z
$H(z)=\frac{b_0+b_1z^{-1}+\cdots+b_mz^{-m}}{1-a_1z^{-1}-\cdots-a_nz^{-n}}$

Figure 10.10: The transfer function of a linear time-invariant system in the Z domain.

steady-state error. The figure 10.9 summarizes these control components, and illustrates the relationships between them.

The table 10.1 summarizes the corresponding control signals. Given the complex nature of the feedback control theory, we do not present a justification for this selection at this time. Instead, a series of separate chapters (11, 12 and 13) will address the design details for each control algorithm. The naming of the signals is done from the perspective of the control theory (e.g. control input, control output, etc.), except for the stochastic algorithm, shown in italics.

Signal	Notation	Description
		Fast control
Input	$w_f(k)$	The number of *fast* upload sessions during the sampling interval k.
Output	$y_f(k) = w_f(k) - u(k)$	The difference between the number of upload sessions and the average number of channel viewers during the sampling interval k.
Reference	$r_f(k) = 0$	The desired value for the output is zero.
Error	$e_f(k) = r_f(k) - y_f(k-1)$	The difference between the reference and the control output.
		Stochastic control
Error	$e_h(k) = u(k) - w_f(k)$	The difference between the number of channel viewers and the number of fast upload sessions during the sampling interval k.
Input	$w_h(k)$	The number of *stochastic* upload sessions during the sampling interval k.
		Slow control
Input	$w_s(k)$	A correction factor for the fast upload sessions, during the sampling interval k, which compensates for the blocking ratio steady-state error generated by the limitations of the fast control algorithm.
Output	$y_s(k) = b(k)$	The blocking ratio.
Reference	$r_s(k) = r_b = \beta$	The desired blocking ratio, administratively set by the P2PTV provider.
Error	$e_s(k) = r_s(k) - y_s(k-1)$	The difference between the reference and the control output.

Table 10.1: The definition of the SCA control signals.

In addition to the time-domain representation, the Z-transform is a powerful tool for the analysis of discrete-time signals and systems. By operating in the Z-domain, the relationship between the input and the output of a linear time-invariant system can be expressed in terms of an algebraic product. The figure 10.10 illustrates a system with an input signal, $x(k)$, and an output $y(k)$. The equivalent Z-domain signals are $X(z)$ and $Y(z)$, where $Y(z) = X(z)H(z)$ and $H(z)$ is the system transfer function.

For a feedback control system, we generally denote the transfer function of a controller by $K(z)$, and the transfer function of the measured system by $G(z)$. By applying these notations, in the figure 10.11, we show the complete architecture of the session coordination algorithm. The

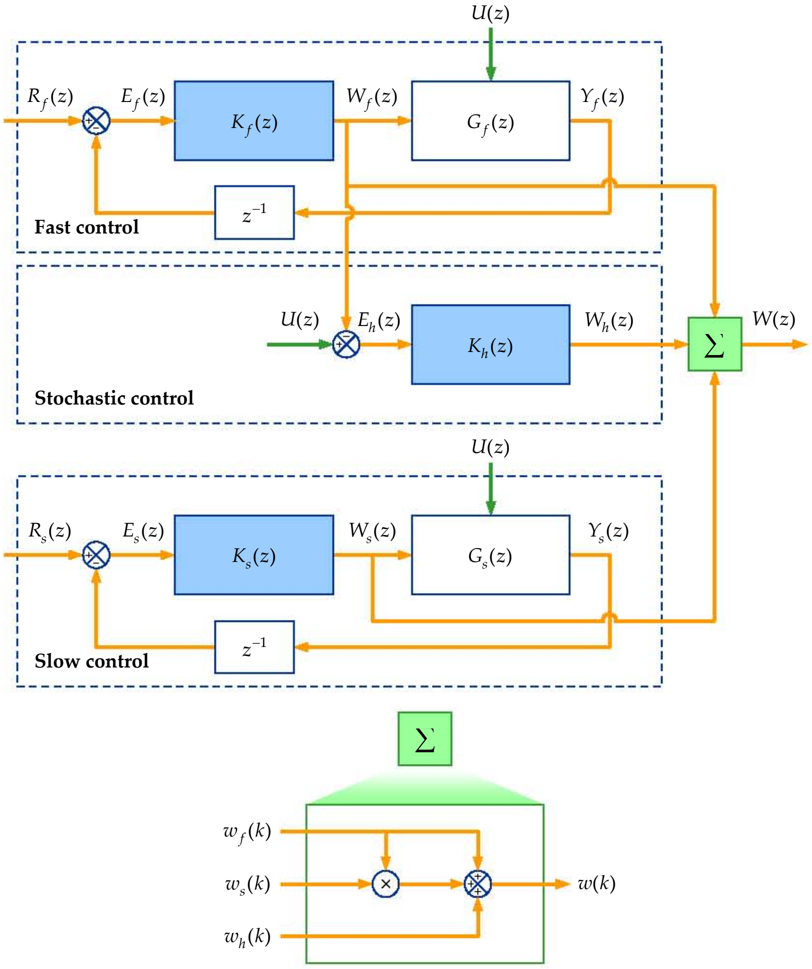


Figure 10.11: The complete architecture of the session coordination algorithm.

figure emphasizes the three main components: fast, stochastic and slow algorithms. The table 10.2 lists the notations used for both the Z-domain signals and transfer functions. By applying the definition of the control inputs, we obtain the total number of upload sessions:

$$w(k) = w_f(k) \cdot (1 + w_s(k)) + w_h(k). \tag{10.5}$$

Solving the design problem of the session coordination algorithm means finding the controller transfer functions $K_f(z)$, $K_h(z)$ and $K_s(z)$ or their time-domain equivalents (when the system is not linear), which maximize the performance in terms of blocking ratio and session utilization. Toward this end, in the following three chapters we take a rigorous approach for each of these, as follows:

- the chapter 11 presents the *fast control* algorithm;
- the chapter 12 presents the *stochastic control* algorithm, and;
- the chapter 13 presents the *slow control* algorithm.

10.4 The Peer Coordination

In contrast to the session coordination function, which estimates the number of active and inactive upload sessions, and implicitly the number of foster peers, the *peer coordination algorithm*, or PCA,

Variable	Fast	Stochastic	Slow
Controller	$K_f(z)$	$K_h(z)$	$K_s(z)$
System	$G_f(z)$	–	$G_s(z)$
Input	$W_f(z) = E_f(z)K_f(z)$	$W_h(z)$	$W_s(z) = E_s(z)K_s(z)$
Output	$Y_f(z) = W_f(z)G_f(z)$	–	$Y_s(z) = W_s(z)G_s(z)$
Reference	$R_f(z) = 0$	–	$R_s(z) = r_b = \beta$
Error	$E_f(z) = R_f(z) - z^{-1}Y_f(z)$	$E_h(z)$	$E_s(z) = R_s(z) - z^{-1}Y_s(z)$

Table 10.2: The control signals and the transfer functions in the Z-domain.

focuses on allocating the peer bandwidth resources to these sessions, such that they accommodate the channel viewer demand.

As shown in the figure 4.16 from the chapter 4, the PCA manages both uploading and downloading sessions, in response to user activity events and control updates from the SCA. Depending on the available network resources, the PCA operates in one of the following scenarios:

- *resource-relaxed*, where the UE peers have sufficient bandwidth to serve all requests, or;
- *resource-constraint*, where the peer bandwidth is appropriated to each channel, proportionally to a given metric.

Normally, we expect the latter situation more common in a real-life deployment. Since solving the session allocation problem, in resource-constraint system, is not a trivial one, we dedicate the chapter 14 to the design of the peer coordination algorithm.

10.5 Conclusions

The architecture of the P2PTV application server, or P2PTV-AS, is divided into three elements: the SIP function, the IPTV function and the P2PTV function. The SIP function constitute the lower layer of the P2PTV-AS, which manage the multimedia sessions from and to the UE peers, via the ISC reference point. Depending on the parties involved and the handling at the P2PTV-AS, the sessions are classified in *terminating*, *originating* and *back-to-back*.

The IPTV function correspond to the functional entities standardized in the TISPAN specifications. The service selection function, or SSF, maintains the list of the available TV channels, which are distributed to the user equipments on the service subscription dialog. The service control function, or SCF, handles the communication with the broadcast server, upon requests received from the user equipments.

The P2PTV function, representing the main focus of the present work, consist of two algorithms: the session coordination and the peer coordination. The session coordination algorithm (SCA) computes, for each TV channel, the number of upload sessions necessary to implement the fast signaling enhancement. The SCA models the P2PTV service as a queuing system, and uses a feedback control loop to adjust the number of sessions to the dynamics of the user activity. The algorithm input is the desired blocking ratio, β, the viewers arrival is considered a disturbance, and the output is the number of upload sessions, w.

By carefully considering the design requirements, as well as the limitation of the blocking ratio metric, we divide the SCA into three control algorithms: fast, stochastic, and slow. The fast algorithm (chapter 11) ensures a fast response to changes in the user activity, at every sampling interval. The stochastic algorithm (chapter 12) uses the stochastic properties of fast algorithm error to drive the blocking ratio to the reference value. Finally, the slow algorithm (chapter 13) corrects any long-term steady-state error.

The peer coordination algorithm (PCA), presented in detail in the chapter 14, accommodates both the active and inactive session requests, by allocating the network resources available to the set of UE peers. These resources are shared among all TV channels proportionally to a performance metric.

Chapter 11

The Fast Control Algorithm

11.1 Overview

The *fast control algorithm*, as part of the session coordination function, evaluates the number of active and inactive upload sessions needed to accommodate the viewer demand on individual TV channels. While the number of active sessions is always linked to the user equipments currently connected, the number of inactive sessions is a prediction of the future necessity that considers the users expected to connect, and the session establishment delay.

To restate the overall objective, the SCA computes the number of upload sessions at the time moment k, denoted by $w(k)$, which minimizes the error between the blocking ratio of the incoming requests, $b(k)$, and the desired reference β, and maximized the session utilization ρ. Unfortunately, the measurement of the blocking ratio is a slow process, as it requires the averaging of the TV channel activity over a larger time interval. Because in practice, the user activity changes much more rapidly, especially at the boundaries between TV programs, we introduce the fast control as the first step in a three stage algorithm.

The main idea is using a control output signal, which we can measure accurately during a sampling interval. Such a metric should be correlated to the blocking ratio and the session utilization, in that a performance improvement in the fast control loop should reflect in a small $b(k)$ and a high $\rho(k)$.

11.1.1 Basic Design

The figure 11.1 illustrates the feedback control loop of the fast control algorithm. We begin our design with the selection of the fast algorithm signals: control input, control output and control error. In the previous chapter, we introduced briefly these signals without a proper justification, in the interest of keeping the presentation as simple as possible. Our choice is influenced by the equation 10.4 describing the blocking ratio, while balancing between blocking and utilization.

We observe that the blocking ratio increases for all arrival instances where the number of upload sessions is smaller than the number of channel users. At the opposite, we have the

Color Key

🟧 Algorithm signal

🟩 External disturbance

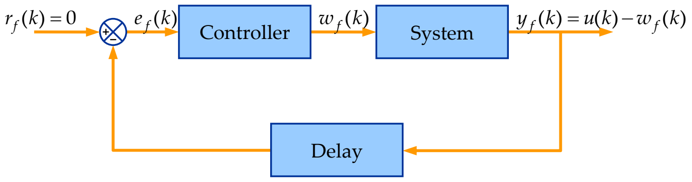


Figure 11.1: The basic design of the fast control algorithm.

utilization decreasing when the number of session is greater than the number of users. We denote by $u(k)$ the average number of users during the interval k, and by $\min\{u(k)\}$ and $\max\{u(k)\}$ the instantaneous minimum and maximum, respectively. Hence, we may conclude:

$$\begin{array}{llll} \text{when} & w(k) < \max\{u(k)\} & \text{then} \;\; b(k) & \text{increases,} \\ \text{when} & w(k) > \min\{u(k)\} & \text{then} \;\; \rho(k) & \text{decreases.} \end{array} \tag{11.1}$$

By choosing the fast control input $w_f(k)$ as the number of upload sessions, we select the fast control output as the difference:

$$y_f(k) = u(k) - w_f(k). \tag{11.2}$$

This selection has several advantages. First, the average number of channel users can be measured easily at every k. Second, the control output is a good indicator of the system performance. A positive output shows that there are more channel viewers than upload sessions, hence a poor blocking ratio; a negative output reflects a poor utilization. The reference signal $r_f(k)$ tunes the balance between b and ρ, and by default is set to zero. Finally, the control loop is linear and allows for an easy design of the controller transfer function.

The control input $w_f(k)$ captures the number of upload sessions computed at the beginning of the sampling interval $k-1$. This approach has the advantage of simplicity, by considering the real session establishment delay. However, the sampling period should be large enough to allow the establishment of the upload sessions, according to the P2PTV SIP signaling procedures, with a reasonable margin. In the figure 11.2(a), we illustrate the delay corresponding to the session establishment process. The steps involved are the following.

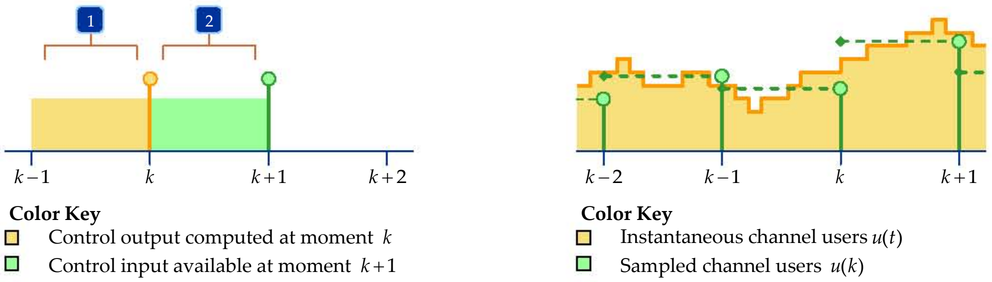
1. During the interval between the samples $k-1$ and k, the SCA fast control loop measures the average number of channel users, $u(k)$. At the moment k, the algorithm computes the control output $y_f(k) = u(k) - w_f(k)$, and the control input for the next sampling moment, $w_f(k+1)$.

2. During the interval between the samples k and $k+1$, the P2PTV-AS begins establishing any sessions, if any, required by the next value of the control input. Our design requires the establishment of these sessions completed within one sampling period[1].

The discrete signal $u(k)$, representing the number of channel viewers, does not sample but averages the instantaneous number of viewers, $u(t)$ over the interval $k-1..k$. The measurement process is shown in the figure 11.2(b), and we have:

$$u(k) = \frac{1}{T}\int_{(k-1)T}^{kT} u(t)\mathrm{d}t, \tag{11.3}$$

[1] The selection of a sampling period of 10 seconds should allow sufficient time for the completion of all signaling procedure. However, the completion of the IMS signaling serves as a constraint when needing smaller sampling intervals.


1
2
k−1
k
k+1
k+2
Color Key
🟨 Control output computed at moment k
🟩 Control input available at moment k+1
k−2
k−1
k
k+1
Color Key
🟨 Instantaneous channel users u(t)
🟩 Sampled channel users u(k)

(a) The delay of the fast control loop.

(b) The measurement of the average number of users, $u(k)$.

Figure 11.2: Signal processing on the fast control loop.

where T is the sampling period.

In the implementation of the P2PTV-AS, computing the average samples is as simple as incrementing the area below the instantaneous curve for every user arrival or departure event, and then dividing by the sampling period for every control clock event. We illustrate briefly this principle in the code sample 11.1.

Code sample 11.1: Computing the control signal $u(k)$.

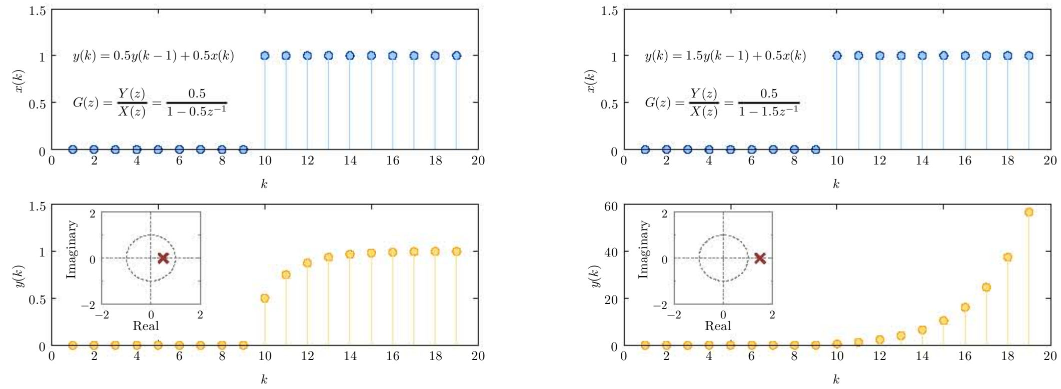
```
% u      - the average number of channel viewers
% U      - the instantaneous number of channel viewers
% t_last - the moment of the last event
% T      - the sampling period

function EventUser(t, U)
    k = floor(t/T) + 1; % Compute the sampling interval
    u(k) = u(k) + U * (t - t_last); % Increment the area
    t_last = t; % Update the moment of the last event
end

function EventControl(k, U)
    u(k) = u(k) + U * (k * T - t_last); % Increment the area
    u(k) = u(k) / T; % Divide by the sampling period
    t_last = k * T; % Update the moment of the last event
end
```

11.1.2 Design Objectives

The rationale of a feedback control system is adjusting the control input according to the deviations of the control output from the desired reference. Its power stems from the feedback path, which brings back information from the controlled system allowing one to correct unexpected disturbances. Unfortunately, it is the same power that makes designing a feedback system a more challenging task.

One of the dangers is that the controller, which converts the control error to the control input, under-reacts or over-reacts to the error changes. In the first case, the control loop is slow responding to disturbances, while in the latter, the output may overshoot beyond the reference, start oscillating or even increase out of control.

In the systems theory the properties characterizing the aforementioned behavior are the *stability*, *settling time*, *steady-state gain* and *maximum overshoot*. It is not our purpose to introduce these notions in greater depth; we believe that the work of Hellerstein et al. represents an excellent overview of the topic (Hellerstein et al., 2004). However, we shall briefly present how to examine this properties from the designing point of view.



(a) Stable system with transfer function pole $p = 0.5$.

(b) Unstable system with transfer function pole $p = 1.5$.

Figure 11.3: The stability of a DLTI system.

Let us consider a discrete, linear and time invariant (DLTI) system of an arbitrary order, with the Z transfer function $G(z)$, where:

$$G(z) = \frac{b_0 + b_1 z^{-1} + \cdots + b_m z^{-m}}{1 - a_1 z^{-1} - \cdots - a_n z^{-n}}. \tag{11.4}$$

Through the factorization of both the numerator and the denominator, we obtain:

$$G(z) = \frac{(z - z_1) \cdots (z - z_m)}{(z - p_1) \cdots (z - p_n)}. \tag{11.5}$$

The roots of the numerator, $z_1 \ldots z_m$, are called the *zeros* of the transfer function, whereas the roots of the denominator, $p_1 \ldots p_n$, are called the *poles*, and may be either real or complex. The roots, and the poles in particular, play an essential role in assessing the system behavior.

In terms of stability, a DLTI system is *bounded-input bounded-output (BIBO) stable* if and only if the poles of the transfer function, are within the unit circle on the complex plane:

$$\text{A DLTI system is BIBO stable} \quad \Longleftrightarrow \quad |p_i| < 1, \quad \forall i = 1..n. \tag{11.6}$$

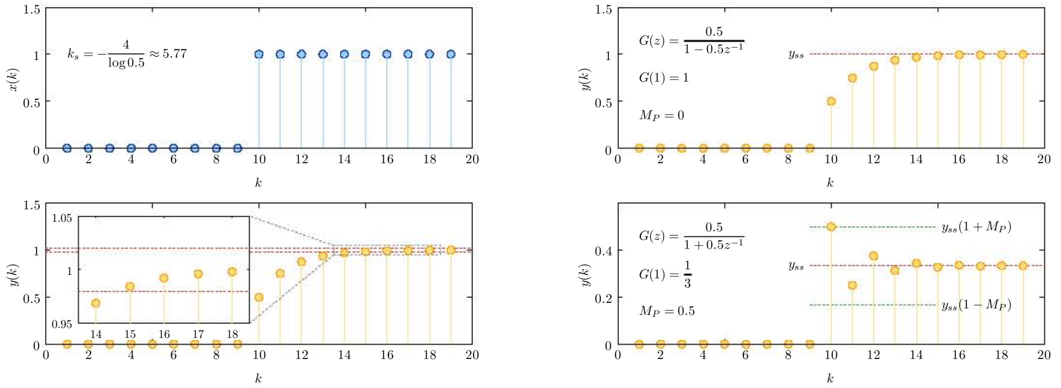
The figure 11.3 compares two simple exponential moving average systems, where the one on the left is stable, with a single pole within the unit circle, and the one on the right unstable, with the pole outside the unit circle. The input signal $x(k)$ is a simple step function. For the stable system, the output, $y(k)$, always converges to a finite steady-state value.

The *settling time*, denoted by k_s, measures the interval, in number of samples, between the change of the input and the moment the output is sufficiently close to the new steady-state value, where a ± 0.02 deviation is the accepted interval. In general, the settling time is influenced by the pole with the largest magnitude (also called dominant pole), and we have:

$$k_s \approx -\frac{4}{\log \max |p_i|} \tag{11.7}$$

The figure 11.4(a) illustrates the settling time, for the case of the previous stable system with a single pole $p_1 = 0.5$.

The *steady-state gain* is the ratio between output and input in steady-state conditions, denoted by x_{ss} and y_{ss}, respectively. For a system characterized by the Z transfer function, the steady state gain is:



(a) The settling time.

(b) The steady state gain and maximum overshoot.

Figure 11.4: The properties of a DLTI system.

$$\frac{y_{ss}}{x_{ss}} = G(1). \tag{11.8}$$

In the figure 11.4(b), we show the steady state gain for two exponential moving average systems, which only differ by the selection of the pole, at $p_1 = 0.5$ and $p_1 = -0.5$, respectively. In both cases, the input signal is the same step function as in previous examples, and we have $x_{ss} = 1$ and $y_{ss} = G(1)$.

Finally, the *maximum overshoot*, denoted by M_P, measures the maximum deviation between the output and its steady-state value, normalized to the steady-state value:

$$M_P = \frac{|y_{max} - y_{ss}|}{|y_{ss}|}, \tag{11.9}$$

where we only consider deviations which are non-monotonic, such as oscillations. As in the case of the settling time, the maximum overshoot is influenced by the transfer function dominant pole, p_i, and we have:

$$M_p \approx \begin{cases} 0 & p_i \in \mathbb{R} \wedge p_i \geq 0, \\ |p_i| & p_i \in \mathbb{R} \wedge p_i < 0, \\ r^{\pi/|\theta|} & p_i \in \mathbb{C} \setminus \mathbb{R} \wedge r = |p_i| \wedge \theta = \arg p_i. \end{cases} \tag{11.10}$$

In the figure 11.4(b), the first (top) system has a real and positive dominant pole, and hence the overshoot is $M_P = 0$. The second (bottom) system has a real and negative pole, and the overshoot is approximative equal to the pole magnitude, with the output oscillating around the steady-state value.

In our design, we consider the previous characteristics of a feedback control system, and we summarize the following desirable properties:

- *stability*, that is the fast control algorithm should not overreact to changes in the control output;
- *accuracy*, having a small settling time k_s, comparable to the sampling interval, and zero or small overshoot;
- *balanced*, that is the algorithm should yield a good compromise between the blocking ratio $b(k)$ and session utilization $\rho(k)$.

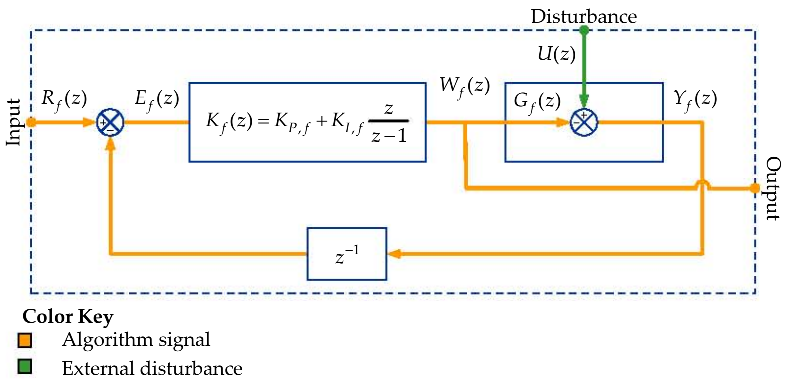
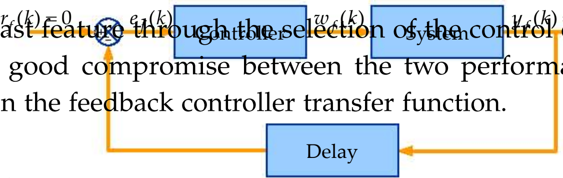


Figure 11.5: The feedback control loop design of the fast control algorithm.

We address the last feature through the selection of the control output and reference signals, which represents a good compromise between the two performance metrics. The first two properties depend on the feedback controller transfer function.

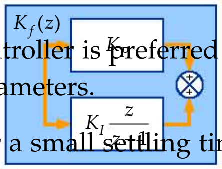


11.2 Controller Design

11.2.1 Feedback Loop

We use a proportional-integral (PI) controller to translate the control error into the number of sessions. The reasons for selecting such a controller, as opposed to a proportional-integral-derivative (PID), include:

- In many circumstances, a PI controller is preferred to a PID type due to simplified analysis and design of the controller parameters.
- In our case, the requirement for a small settling time would imply a very small derivative component since the difference between consecutive samples of the number of users is representative of the actual user trend. Therefore, using a larger derivative component could very easily introduce instability in the closed-loop system.



- A method of counteracting the spurious variation of the channel users is to introduce a smoothing filter. However, the slower response (delay) of such a filter would defeat the purpose of the fast response desired through the derivative component.

Later in this chapter, we verify whether our selection of a PI controller was reasonable, by performing an empirical study of our algorithm on existing workload data using a PID controller instead of the PI variation. Our results show that in order to achieve a small control error and a short settling time, the contribution of the derivative component must be indeed very small.

The figure 11.5 illustrates the feedback loop and the PI controller used by the fast algorithm. In time-domain, the control input can be written as:

$$w_f(k) = K_{P,f} e_f(k) + K_{I,f} \sum_{i=0}^{k} e_f(i), \tag{11.11}$$

where the first component is proportional to control error, weighted by the proportional coefficient $K_{P,f}$, and the last is the integral of the control error, weighted by the integral coefficient $K_{I,f}$. By rewriting the equation in difference form, $w_f(k) - w_f(k-1)$, we have:

$$w_f(k) = w_f(k-1) + K_{P,f}\Big(e_f(k) - e_f(k-1)\Big) + K_{I,f} e_f(k), \tag{11.12}$$

and then, by applying the Z transform, we obtain:

$$W_f(z) = E_f(z)\left(K_{P,f} + K_{I,f}\frac{z}{z-1}\right), \tag{11.13}$$

where $K_f(z)$ is the controller transfer function:

$$K_f(z) = K_{P,f} + K_{I,f}\frac{z}{z-1}. \tag{11.14}$$

Since both the reference and disturbance signals are known, we can write the closed-loop transfer function between the control input (i.e. the output of the fast algorithm) and the disturbance. We have:

$$H_R(z) = \frac{W_f(z)}{U(z)} = \frac{1}{1 - z^{-1}K_f(z)} = \frac{z(z-1)}{z^2 - \left(K_{P,f} + K_{I,f} + 1\right)z + K_{P,f}}. \tag{11.15}$$

11.2.2 The PI Controller

To solve the coefficients of the PI controller, considering the aspects on stability, settling-time and output overshoot, we apply the root-locus method (Hellerstein et al., 2004). This approach is generally applicable when the denominator or the *characteristic equation* of the closed-loop transfer function can be written in the form:

$$1 + \kappa H(z) = 0, \tag{11.16}$$

where $\kappa > 0$ represents the *feedback gain* of the system. In these settings, the root locus shows the trajectories of the closed-loop poles (that is the roots of the characteristic equation, hence the name of the method), denoted by $p_{c,1}$ and $p_{c,2}$, when the feedback gain κ goes from zero to infinity.

To this end, we can rewrite the closed-loop characteristic equation as:

$$\begin{aligned} 1 - z^{-1}\left(K_{P,f} + K_{I,f}\frac{z}{z-1}\right) &= 1 - (K_{P,f} + K_{I,f})\frac{z - \dfrac{K_{P,f}}{K_{P,f} + K_{I,f}}}{z(z-1)} \\ &= 1 + \kappa\frac{z - z_0}{z(z-1)}, \end{aligned} \tag{11.17}$$

where:

$$\kappa = -(K_{P,f} + K_{I,f}) \tag{11.18}$$

is the feedback gain, and:

$$z_0 = \frac{K_{P,f}}{K_{P,f} + K_{I,f}} \tag{11.19}$$

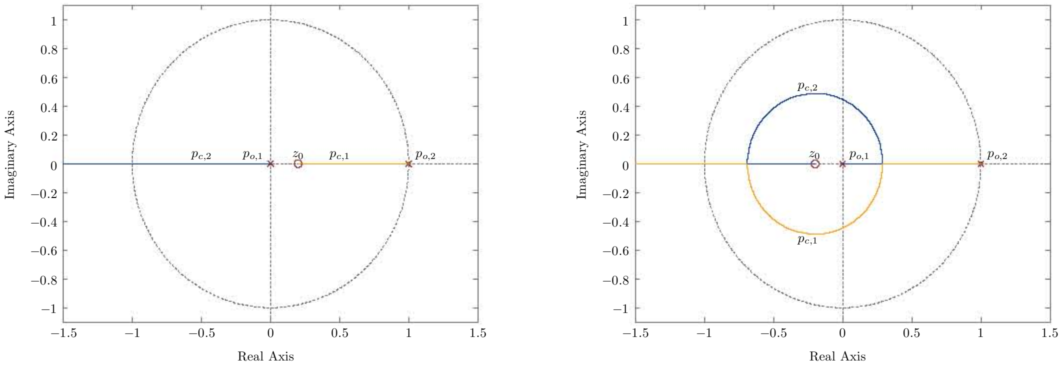
is the open-loop zero.

By applying the previous change of variables, the characteristic equation of the closed-loop system becomes:

$$z^2 + (\kappa - 1)z - \kappa z_0 = 0, \tag{11.20}$$

and the closed-loop poles can be written as:

$$p_{c,(1,2)} = \frac{1}{2}(1-\kappa) \pm \frac{1}{2}\sqrt{(\kappa-1)^2 + 4\kappa z_0}. \tag{11.21}$$



(a) $0 < z_0 < 1$

(b) $-\infty < z_0 < 0$

Figure 11.6: The root loci of the closed-loop characteristic for the second-order system.

11.2.2.1 Second-Order System

First, let us assume that the open-loop zero is not equal to any of the open-loop poles. To determine the root locus, we begin by choosing a value for the zero z_0. Considering that the poles of the open-loop system are $p_{o,1} = 0$ and $p_{o,2} = 1$ (the roots of the denominator of $K_f(z)$), we have the following options[2]:

- A real positive zero at the between the open-loop poles at $p_{o,1} = 0$ and $p_{o,2} = 1$, that is $0 < z_0 < 1$. In this case, both closed-loop poles are real, and one always inside the unit circle.
- A real negative zero at the left of the open-loop pole at $p_{o,1} = 0$, that is $z_0 < 0$. In this case, the closed-loop poles can be either real or complex.

Considering the feedback gain, κ, going from the zero to the infinity, we obtain the following limits for the closed-loop poles[3]:

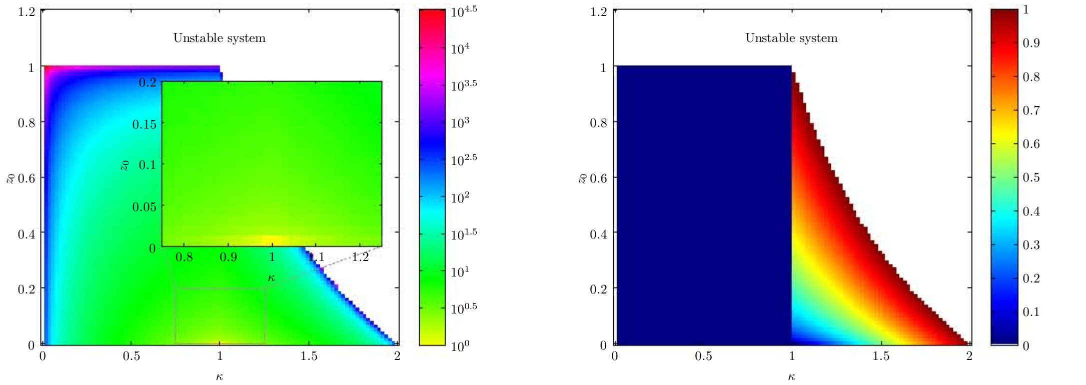
$$\begin{aligned} \lim_{\kappa \to 0} p_{c,1} &= 0 \quad \lim_{\kappa \to \infty} p_{c,1} = z_0, \\ \lim_{\kappa \to 0} p_{c,2} &= 0 \quad \lim_{\kappa \to \infty} p_{c,2} = -\infty. \end{aligned} \tag{11.22}$$

With this result, the figures 11.6(a) and 11.6(b) show the root loci of the closed-loop system. When the open-loop zero is positive, $0 < z_0 < 1$, both poles are real, and one is always bounded by the unitary interval. In addition, one pole is positive, while the second is negative for any gain κ. When the zero is negative, $z_0 < 0$, the poles may be either real or complex, and depending on the value of the zero and gain, may be inside or outside the unit circle. We remind the reader that complex poles are undesirable, because they always generate an overshoot in the control output. Although there exists a set of real solutions for a negative open-loop zero, we prefer the option of a positive z_0, which offers a higher degree of freedom for the range of the feedback gain.

For a positive open-loop zero, $0 < z_0 < 1$, and an arbitrary feedback gain, κ, we address, one at a time, the properties of the feedback system. The first, *stability*, requires the closed-loop system poles within the unit circle of the complex plain. Because the constraint applies only to $p_{c,2}$, and the pole is always negative, the fast control algorithm is stable if and only if:

[2] We note that there exists a third option, where the zero is placed on the real axis at the right of the open-loop pole $p_{o,2} = 1$, that is $z_0 > 1$. However, for such a selection one closed-loop pole is outside the unit circle, and the system is always unstable.

[3] The appendix C.1 presents the proof.



(a) The settling time, k_s, over the stability region.

(b) The maximum overshoot, M_P, over the stability region.

Figure 11.7: The characteristics of the closed-loop system.

$$p_{c,2} > -1. \tag{11.23}$$

By replacing with the pole expression, as function of z_0 and κ, we obtain:

$$\begin{aligned} -\frac{1}{2}(\kappa-1) - \frac{1}{2}\sqrt{(\kappa-1)^2 + 4\kappa z_0} &< -1 \\ (\kappa-1) - \sqrt{(\kappa-1)^2 + 4\kappa z_0} &< 2 \\ \sqrt{(\kappa-1)^2 + 4\kappa z_0} &< 3-\kappa \\ (\kappa-1)^2 + 4\kappa z_0 &< (3-\kappa)^2 \\ \kappa &< \frac{2}{z_0+1}. \end{aligned} \tag{11.24}$$

The *settling time* is approximated by the pole with the largest magnitude. In the figure 11.7(a), we examine it for the whole stability region, $0 < z_0 < 1$ and $0 < \kappa < 2/(z_0+1)$, and we find the minimum settling time in the region of an unitary feedback gain and near-zero z_0. This result is not surprising, because when $z_0 = 0$, it cancels the open-loop pole $p_{o,1} = 0$, transforming the feedback loop in a first order system. Such a system has a single closed-loop pole, which is zero when $\kappa = 1$.

The *maximum overshoot* shows a similar trend, in the figure 11.7(b), with the notable exception that for a sub-unitary gain, the positive pole is dominant, and M_P is always zero. Therefore, we choose $z_0 \approxeq 0$ and $\kappa = 1$.

11.2.2.2 First-Order System

At limit, we may select an open-loop zero that overlaps to the open-loop zero, $z_0 = p_{o,1} = 0$. According to the definition of z_0, we obtain $K_{P,f} = 0$, and the controller becomes purely integral. The closed-loop transfer function is:

$$H_R(z) = \frac{1}{1 - K_{I,f}\dfrac{1}{z-1}} = \frac{z-1}{z-1-K_{I,f}}, \tag{11.25}$$

and by following a similar principle, the feedback gain is $\kappa = -K_{I,f}$. There exists a single closed-loop pole $p_{c,1} = 1 - \kappa$, and like in the situation of the second-order system, the settling time and maximum overshoot are both minimum when $\kappa = 1$. With these observations in mind, the parameters of the PI controller are:

$$\begin{aligned} K_{P,f} &= 0, \\ K_{I,f} &= -1. \end{aligned} \tag{11.26}$$

11.3 Performance Evaluation

We ascertain the performance of the fast control algorithm by simulating a P2PTV system, using the realistic workload model presented in the part III of this thesis. The response of the control system versus the user activity intensity, measured as channel arrivals and departures, is of particular interest, because the fixed sampling period, of 10 seconds, cannot compensate activity dynamics happening at smaller time scales. To this end, we selected three equivalent scenarios with 10 000, 50 000 and 100 000 active subscribers. Increasing the number of viewers allows us to determine the scalability of the P2PTV service. In our experiments, the size of user population was limited by the computing power available to perform the simulations.

The performance metrics are the fast control output, $y_f(k)$, the blocking ratio, $b(k)$ and session utilization, $\rho(k)$. Although the fast control loop does not directly drive the last two metrics to the desired reference, by including them allows us to examine whether the choice of the control output was appropriate.

In addition to testing our algorithm in the aforementioned standard conditions, we extend the evaluation to answer the following questions.

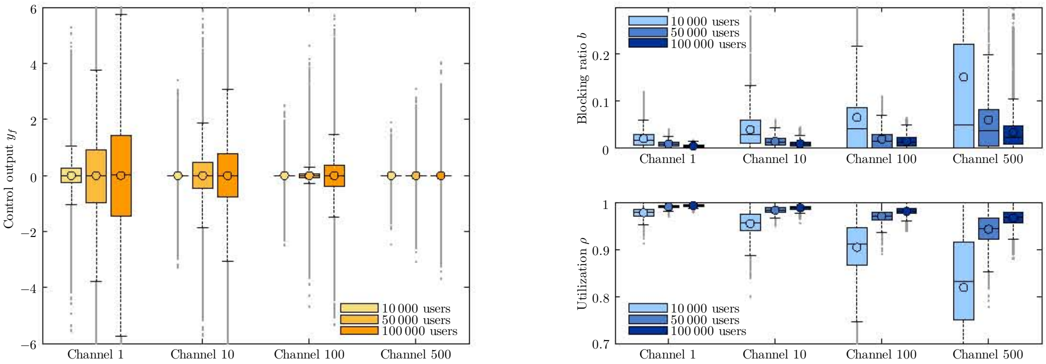
- Is the integral controller appropriate, or would a PID controller have performed better?
- How does the algorithm respond to a activity workload outside the envelope of our present model?
- How does the sampling period affect performance?

11.3.1 Response to Normal User Activity

We begin with an analysis of the control loop response to the normal TV user activity, and the default configuration parameters. Since the performance characterizes individual TV channels, we restrict our results to four TV channels spread, roughly proportional, along the popularity spectrum of 700 channels: 1, 10, 100, and 500. To illustrate the data, we used a modified box plot that, in addition, displays the sample mean[4].

First, in the figure 11.8(a), we examine the distribution of the control output, $y_f(k)$. Ideally, the control output matches the reference value, set to zero, balancing between the blocking ratio and the session utilization. In reality, due to randomness of the channel arrivals, the fast loop either over- or under-estimates the number of upload sessions needed to accommodate the demand. The error is greater with increasing user arrival and departure rate, observed for both popular channels and when the number of TV subscribers is larger.

[4] In our modified box plot the rectangular box illustrates the lower quartile (the 25th percentile or Q_1) and the upper quartile (the 75th percentile or Q_3) of the population. The middle line represents the population median (Q_2), and the whiskers show the minimum and maximum samples that are not considered outliers ($Q_1 - 1.5\text{IRQ}$ and $Q_3 + 1.5\text{IRQ}$, respectively, where the inter-quartile range is $\text{IRQ} = Q_3 - Q_1$). The outliers are plotted individually. In addition, the plot illustrates with a small circle the sample mean, for situations where the median of the population is not representative.



(a) The control output, y_f.

(b) The blocking ratio, b, and upload session utilization, ρ.

Figure 11.8: The performance of the fast control algorithm for normal user activity.

A positive deviation of the control output affects the blocking ratio, b, whereas a negative deviation ultimately impacts the utilization, ρ. Because the channel arrival and departure rates are approximately equal throughout the observation period, the mean control output is practically zero. In the worst scenario, of the most popular channel for a deployment with 100 000 subscribers, the inter-quartile range is ± 3 sessions, very small compared to the number of channel viewers, in the thousands.

In relative terms, the figure 11.8(b) shows the blocking ratio and the utilization. With this result, it becomes obvious that, compared to the previous result, the performance is skewed in the favor of popular channels. We observe that by using the fast algorithm alone, which is only a coarse driver for the b and ρ metrics, is sufficient to obtain a reasonably small blocking and high utilization. In addition, these shows that our algorithm scales well with the size of the subscriber base.

Unfortunately, these results also reflect that the fast loop alone is insufficient to keep the blocking ratio near the desired reference, therefore justifying the need for the stochastic and the slow algorithms. Unpopular TV channels are at a particular disadvantage, where even small control errors lead to significantly lower performance in relative terms.

11.3.2 Assessment of a PID Controller and Coefficient Validation

To address these limitations, first, we need to consider whether the feedback controller is a bottleneck factor of our design. Initially, we selected a PI controller as opposed to a more flexible PID design. This choice was based on the easier determination of controller coefficients, and our informed intuition that the derivative component is of limited efficiency, given the characteristics of the TV channel arrival process.

In order to set aside these concerns, we propose a numeric evaluation of an equivalent PID controller, starting from the known coefficients. The approach is very simple: we let the PID coefficients, $(K_{P,f}, K_{I,f}, K_{D,f})$ take values within a selected subset of $\mathbb{R}_- \times \mathbb{R}_- \times \mathbb{R}_-$[5]. For every triplet, we compute the mean blocking ratio and utilization, ignoring the results whenever the

[5] The previous computation of the controller coefficients relied on the Z transfer function. However, a common sense approach should determine that, in principle, the coefficients must be negative. The control error, $e_f(k)$, represents the difference between the number of upload sessions and average number of viewers. Hence, when the error is positive, there are more sessions that viewers; when the error is negative there are less sessions than viewers. To minimize blocking and maximize utilization, the fast loop should decrease the number of sessions in the former case, and increase it in the latter. This rule stands, whether the number of sessions changes proportionally to the error signal ($K_{P,f}$), to the area under the error signal ($K_{I,f}$), or the slope of the error signal ($K_{D,f}$).

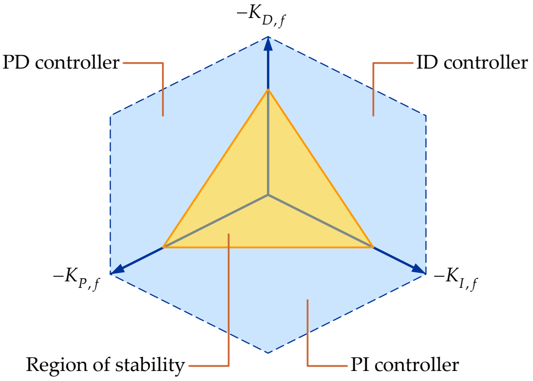
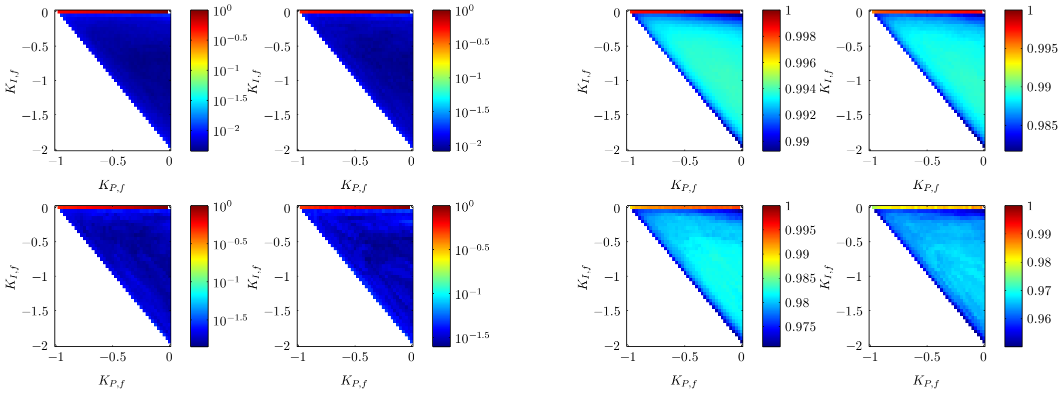


Figure 11.9: The region of stability for a PID controller.



(a) The mean blocking ratio, $\bar{b}$.

(b) The mean utilization, $\bar{\rho}$.

Figure 11.10: The impact of the PI controller parameters, $K_{P,f}$ and $K_{I,f}$, on the performance of the fast algorithm for TV channels 1, 10, 100 and 500 (in clock-wise order from top-left).

closed-loop system is unstable.

The transfer function of the PID controller is:

$$K'_f(z) = K_{P,f} + K_{I,f}\frac{z}{z-1} + K_{D,f}\frac{z-1}{z}. \tag{11.27}$$

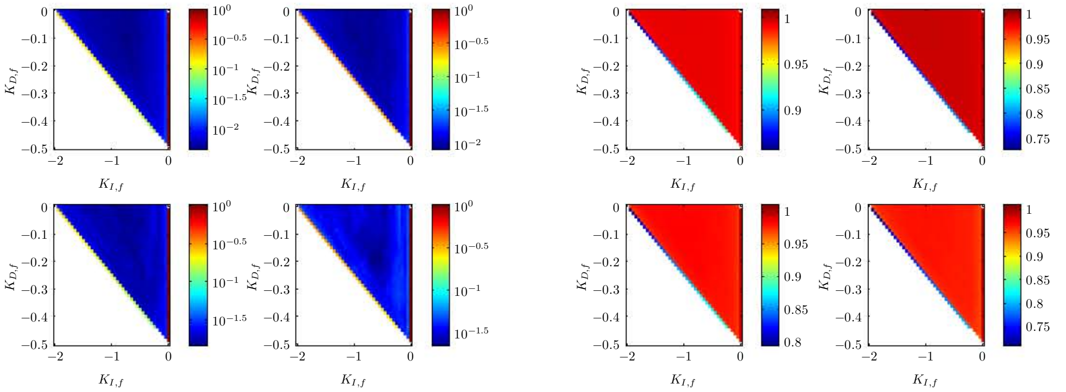
By writing the transfer function of closed-loop system, we obtain the characteristic equation:

$$z^3 - (K_{P,f} + K_{I,f} + K_{D,f} + 1)z^2 + (K_{P,f} + 2K_{D,f})z - K_{D,f} = 0, \tag{11.28}$$

where the roots are the corresponding closed-loop poles, $p_{c,1}$, $p_{c,2}$ and $p_{c,3}$. Since the system is stable, if and only if all poles take values within the unit circle of the complex plane, in the figure 11.9, we illustrate the region of stability of the control loop.

Due to the difficulty of visualizing three-dimensional data, we select two representative sections, where $K_{D,f} = 0$ and $K_{P,f} = 0$, respectively. The figure 11.10 shows the first section, corresponding to the PI controller and where the white area filters the region of instability, delimited by the diagonal and the top $K_{I,f} = 0$. This results confirms our analytical computation of a $(K_{P,f} = 0, K_{I,f} = -1)$ coefficient set at the maximum distance from the stability boundary. The integral component is essential to the elimination of the steady-state error, and its elimination determines a poor response, regardless of the proportional coefficient.

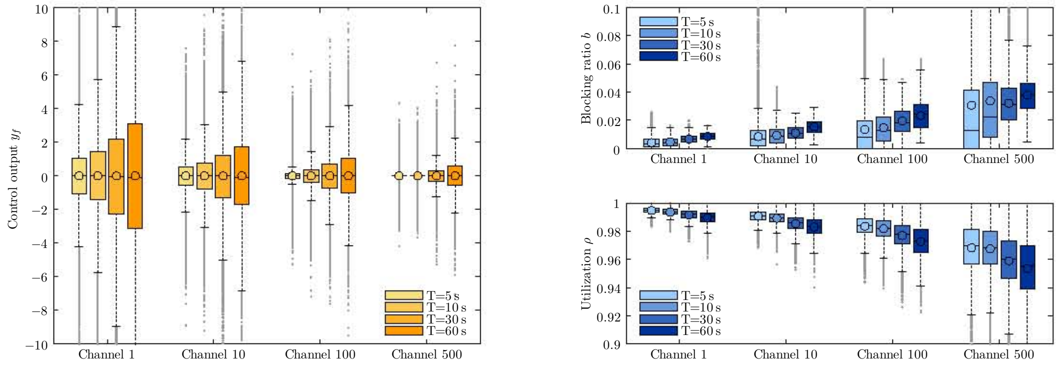
The figure 11.11 illustrates the integral-derivative (ID) plane, where $K_{P,f} = 0$. Similarly, we



(a) The mean blocking ratio, $\bar{b}$.

(b) The mean utilization, $\bar{\rho}$.

Figure 11.11: The impact of the ID controller parameters, $K_{I,f}$ and $K_{D,f}$, on the performance of the fast algorithm for TV channels 1, 10, 100 and 500 (in clock-wise order from top-left).



(a) The control output, y_f.

(b) The blocking ratio, b, and upload session utilization, ρ.

Figure 11.12: The impact of the sampling period on the performance of the fast control algorithm.

obtain the best performance for the pair of coefficients situated as the maximum distance from the stability boundary, delimited by the diagonal and the left $K_{I,f} = 0$. Within this region, increasing the derivative contribution has an insignificant effect on improving the performance metrics, and we conclude that the simple PI design is sufficient for the operation of the fast control loop.

11.3.3 Impact of the Sampling Period

The root cause of the control error distribution is the variation of the channel user activity around the control input computed at the beginning of the sampling interval. As seen earlier, in the figure 11.2(b), the error increases with the TV channel popularity and the number of subscribers in the system. In this context, the sampling period plays a key role, because a smaller interval enables a faster reaction to the channel activity dynamics.

To this end, we analyze the performance of the fast control algorithm versus the length of the sampling period, T, of 5, 10, 30 and 60 seconds, respectively, for 100 000 subscribers. We do not tune T below the 5 second threshold, due to the requirement that the IMS session signaling completes within the current interval. The resulting performance metrics, illustrated in the figure 11.12, confirm that a smaller T has a positive effect on the enhancement of the control error, blocking ratio and utilization.

Although changing the sampling interval is a simple method to adjust the algorithm performance, the effectiveness is limited. For instance, although halving T from 10 to 5 seconds doubles the control effort, there is only a small change in the blocking ratio and utilization. This reinforces our previous assessment that the fast loop alone yields only a modest blocking ratio, and the improvements of the stochastic and slow control are necessary.

11.4 Conclusions

The *fast control loop* is the first step of the session coordination algorithm, to compute the number of upload sessions that, efficiently, accommodate the TV channel demand. Although the main requirement of the SCA is to maintain a blocking ratio close to a desired value β, the primary objective of the fast algorithm is to ensure a speedy response to the user arrival variations.

The algorithm meets this goal, by measuring the difference between the number of channel viewers and upload sessions. The advantage of this control output is twofold: it is easy to measure at every sampling interval, and it is related to both the blocking ratio and the session utilization. The reference value adjusts the balance between the two metrics, such that a positive reference favors a higher utilization and a negative reference results in a smaller blocking ratio.

To compute the control input, which represents the number of upload session, we use a proportional-integral controller. This controller structure is preferred for the simpler design, and a good response to noisy user disturbance. We determine the controller coefficients with the root-locus method, which facilitates the selection of the closed-loop poles considering the stability, settling-time and overshoot requirements.

We assess the performance of the fast control algorithm through large-scale simulations. First, we measure the impact of channel popularity and the number of P2PTV subscribers. In principle, the algorithm maintains the mean control error at zero, however the variance of the data is proportional to the number of channel viewers. In the relative terms however, the blocking ratio and the utilization are worse for unpopular TV channels, which have fewer viewers to begin with.

The selection of the sampling interval is instrumental for the performance of the fast control loop, where a smaller sampling period responds better to the changes in the user activity. Because the interval size is bounded by the session establishment delay, and to a lesser extent by computing power of the P2PTV-AS, there exists a performance limit to the fast control algorithm, which depends on the absolute channel popularity.

Chapter 12

The Stochastic Control Algorithm

12.1 Overview

The *fast control algorithm* adjusts the number of upload sessions, $w_f(k)$, such that the mean error to the average number of users, $e_f(k) = u(k) - w_f(k)$, is kept around the zero reference. However, the main drawback is that the control actions are taken only at discrete moments of time, and therefore any sudden change in the number of channel users cannot be compensated until the beginning of the next sample. For this reason, the performance is bounded by the popularity of the TV channel and the size of the sampling interval.

Since the key of the problem lies in the reaction speed of the control algorithm, a possible solution is to increase the sampling frequency during periods of increased channel activity, and we showed in the section 11.3.3 that smaller samples have a benefic effect on the system performance. On the down side, a higher sampling frequency increases the application server workload as the number of upload sessions must be adjusted more often. Moreover, as we pointed out before, there is a practical lower limit on the sampling period that depends on the session setup delay.

For these reasons, in this chapter, we take a different approach in improving the performance of the fast algorithm. This approach explores the stochastic characteristics of the both the user activity and the behavior of the fast algorithm, and we call it a *stochastic control algorithm*. The stochastic algorithm does to interfere with fast one; instead it acts as an additional enhancement layer by adjusting the number of upload sessions.

12.1.1 Basic Design

The idea behind the algorithm is to measure during each sample the difference between the number of users, $u(k)$, and the number of upload sessions computed by the fast algorithm, $w_f(k)$. We call this difference the stochastic error, defined as:

$$e_h(k) = u(k) - w_f(k), \tag{12.1}$$

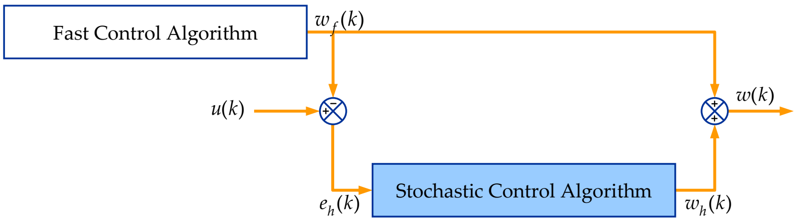


Figure 12.1: The basic design of the stochastic control algorithm.

and it depends on the random user activity during the sampling period that cannot be compensated with the fast approach. By analyzing the stochastic error, the stochastic algorithm generates in real-time a probability model of the error signal. This model opens in turn the possibility to estimate the additional number of upload sessions necessary to have the blocking ratio smaller than or equal to the desired reference value β. The output generated by the stochastic algorithm, $w_h(k)$, is added to the number of sessions already determined by the fast algorithm. The schematic diagram, from the figure 12.1, illustrates this approach.

In order to explain the principle of our approach, let us assume the stochastic error follows a known probability distribution. This supposition entails that given its cumulative distribution function, $F_h(x)$, the probability that the stochastic error is larger than x is:

$$\Pr(e_h(k) > x) = 1 - F_h(x). \tag{12.2}$$

However, in practice we would like to have a blocking ratio, $b(k)$, smaller than a desired reference value β, that is the probability of having an actual difference between user and sessions greater than zero is less than ϵ. At the limit, we prefer:

$$\Pr(u(k) - w(k) > 0) = \beta. \tag{12.3}$$

From the definition of the stochastic error, we can see that the blocking ratio condition is satisfied if to the number of upload sessions computed by the fast algorithm, $w_f(k)$, we add or subtract an extra term denoted by $w_h(k)$ that is the output of the stochastic algorithm. To compute $w_h(k)$, we consider the blocking ratio condition:

$$\begin{aligned} \Pr(u(k) - w(k) > 0) &= \Pr\Big(u(k) - (w_f(k) + w_h(k)) > 0\Big) \\ &= \Pr(u(k) - w_f(k) > w_h(k)) \\ &= \Pr(e_h(k) > w_h(k)) = \beta. \end{aligned} \tag{12.4}$$

Therefore, if the cumulative distribution function of the stochastic error exists and is known, the output of the stochastic algorithm is:

$$w_h(k) = F_h^{-1}(1 - \beta). \tag{12.5}$$

The dynamic term of the last equation is the probability distribution of the stochastic error, that is the expression or the parameters of its CDF. Hence, our stochastic algorithm, using a pool of representative samples, constantly estimated a probability distribution that fits well the measured error.

To demonstrate the principle, let us assume that the stochastic error follows a normal distribution with a zero mean, $\mu_h = 0$, and a determined standard deviation $\sigma_h = 3$, and we require a reference blocking ratio $\beta = 0.05$ for illustrative purposes. In these circumstances, the output

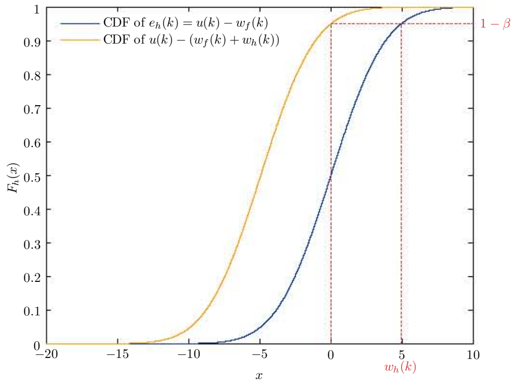


Figure 12.2: The design principle of the stochastic algorithm.

of the stochastic control algorithm should be $w_h(k) = \mathcal{N}^{-1}(\mu_h, \sigma_h) \approx 4.9$, meaning that in order to achieve the reference blocking ratio, given the random nature of the user activity, we should allocate five additional upload sessions to service user demand. The example is depicted in the figure 12.2, where we can see intuitively that the effect of the additional number of upload session is shifting the CDF of the new difference.

12.1.2 Design Objectives

The idea behind the algorithm is simple and straightforward. However, the real challenge is finding the probability distribution that describes the stochastic error, and doing so efficiently and with a reasonable degree of confidence. Toward this end, we summarize the following desirable properties of our algorithm:

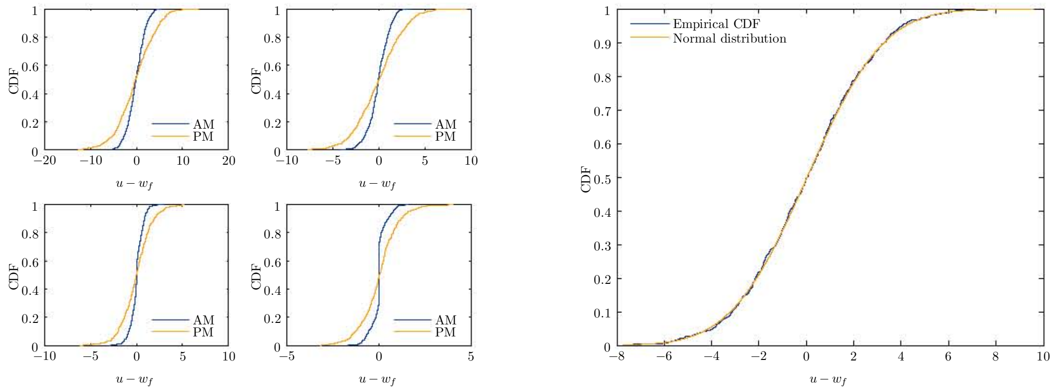
- *accurate*, that is the fit of the stochastic error should determine a CDF that reduces the blocking ratio;
- *efficient*, that is the fitting algorithm can be easily executed for every new error sample, and;
- *historical*, that is newer error samples are give a higher impact that older ones.

12.2 Analysis of Stochastic Properties

12.2.1 Static Properties

We address these design goals by starting with an analysis of the stochastic error resulting from typical user activity. The rationale behind this approach is that if the typical error signal is similar to a well-known probability distribution, then we only have to determine the distributions parameters and the goodness-of-fit. Otherwise, in situations where the error samples do not match any known probability distributions, we would have to resort to empirical methods that are more computationally expensive.

We consider the stochastic error, $w_h(k)$, based on our synthetic activity of 100 000 users and for the same four TV channels spread across the popularity spectrum: 1, 10, 100, and 500. We choose a representative day of the week, and during this day, we examine the error signal during two hours of the morning and the evening time frame: 8:00am–10:00am and 8:00pm–10:00pm. With a sampling period of 10 seconds, the error signal has 720 samples for each TV channel and during each time window.



(a) CDF of the difference between number of users and upload sessions, for TV channels 1, 10, 100, and 500, during AM and PM user activity.

(b) Comparison of the difference between number of users and upload sessions with a normal distribution, for TV channel 10 during PM user activity.

Figure 12.3: Analysis of the stochastic error.

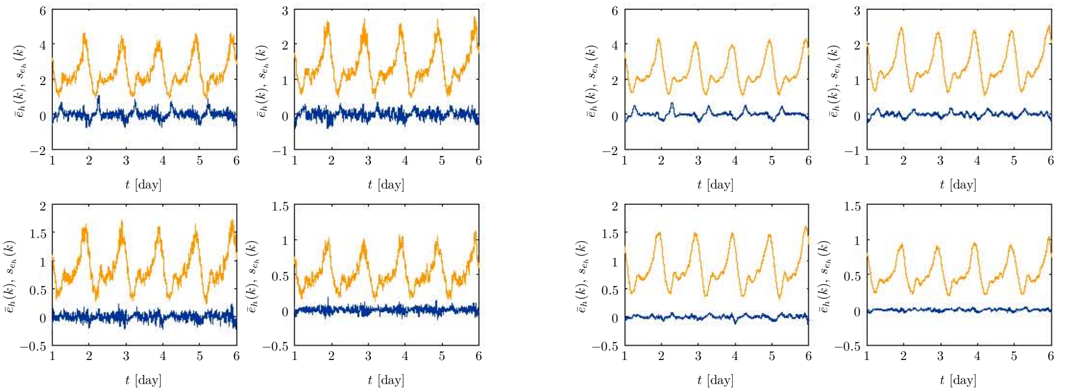
The resulting CDF, illustrated in the figure 12.3(a), offers some unique insights in the possible nature of the stochastic error. First and foremost, the resulting probability distribution is strikingly similar to a normal distribution, especially for popular TV channels or during higher than average (evening) user demand. To illustrate this, in the figure 12.3(b), we compare the CDF of the stochastic error for channel 10 (a popular channel) during the morning with the CDF its corresponding normal distribution, where its parameters (the mean and variance) have been computed from the error data. Second, we observe that the fit to a continuous distribution is less likely for unpopular TV channels (e.g. channel 500), and especially during periods of less intensive user activity such as mornings. This finding suggests that in these situations it will be difficult to fit the sample data to any of the well-known probability distributions, and an empirical method is better suited.

This result is not at all surprising. From the analysis of the user activity, discussed in the chapter 7, we know that the user arrival rate includes a normally-distributed component, which is transferred to arrivals on individual channels proportional to the channel popularity. For popular channels, with a large number of users tuning in during each sample, the fast algorithm compensates poorly this stochastic component, which is transferred as a stochastic error. However, the number of arrivals for unpopular channels is smaller, and therefore better controlled by the fast control loop. For instance, we observe this effect in the morning probability distribution of the channel 500 (an unpopular TV channel), where the CDF of the stochastic error is flattened significantly around zero.

12.2.2 Dynamic Properties

In addition to the shape of the cumulative distribution function, the results from the figure 12.3(a) emphasizes that the stochastic error is characterized by a variance that changes with the time of day. In order to characterize the dynamic properties, we determine the *local* sample mean and variance of the stochastic error.

Computing these parameters raises an important design question: what are the local mean and variance? Our intention is to characterize the probability distribution considering only the short-term history of the system. We approach the problem by considering a fixed-size sliding windowing method, where the sample mean and unbiased sample variance of the stochastic error at the sampling moment k are computed over the last n_w samples, n_w being the window size:



(a) $n_w = 180$ (b) $n_w = 720$

Figure 12.4: The local sample mean ($\bar{e}_h(k)$) and standard deviation ($s_{e_h}(k)$) of the stochastic error. The data represents the user activity on a popular TV channel (channel 10).

$$\begin{aligned} \bar{e}_h(k) &= \frac{1}{n_w} \sum_{i=k-n_w+1}^{k} e_h(k), \\ s^2_{e_h}(k) &= \frac{1}{n_w - 1} \sum_{i=k-n_w+1}^{k} \left(e_h(k) - \bar{e}_h(k)\right)^2 . \end{aligned} \tag{12.6}$$

The selection of sliding window size n_w considers the compromise between a small window that increases the reaction speed with the dynamics of the stochastic error, and a large window that ensures that the computed mean and variance are statistically significant. By considering a sampling period of 10 seconds, we conjecture that a window of size between 30 minutes and 2 hours is an acceptable compromise. Such a window size contains between 180 and 720 data samples of the stochastic error[1], while we can consider the average user activity relatively stable.

By measuring the local mean and variance (or standard deviation) on a given TV channel with two different window sizes, we confirm that the main the window size is proportional to the degree of smoothing applied to these two variables. This result is visible in the figures 12.4(a) and 12.4(b). For our implementation, we select the middle value $n_w = 360$ samples, corresponding to the sliding window of one hour.

12.3 Fitting Algorithm

The probability distribution describing the stochastic error may be approximated with a normal distribution. However, the degree of confidence behind this approximation depends on the channel popularity and level of user activity. We showed previously, that the approximation is good when the user arrival rate is much higher than the sampling frequency, that is popular channels and/or evening hours. However, the fit is poor during the hours of low user activity or for unpopular TV channels. In these circumstances, the CDF presents a strong discontinuity around zero due to the fast control algorithm.

By considering the observed effects, we propose using two distributions that approximate the stochastic error:

[1] The size of the 95th percentile confidence interval around the sample mean for the proposed window size will be between 0.29σ and 0.15σ, respectively, σ being the standard deviation.

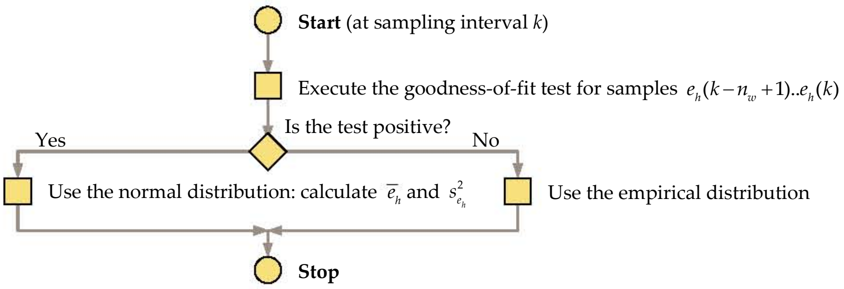

Start (at sampling interval k)
Execute the goodness-of-fit test for samples $e_h(k-n_w+1)..e_h(k)$
Is the test positive?
Yes
No
Use the normal distribution: calculate $\overline{e}_h$ and $s^2_{e_h}$
Use the empirical distribution
Stop

Figure 12.5: The selection algorithm of the probability distribution.

- a *normal distribution* for popular channels and/or high user activity, and;
- an *empirical distribution* for unpopular channels and/or low user activity.

The fitting algorithm for these probability distributions is applied to the set of windowed samples, that is at the sampling moment k, the distributions consider only the stochastic error samples from the interval $[k - n_w + 1..k]$. As shown previously, this approach takes reasonably into account the dynamic properties of the error signal.

The reason for choosing two distributions instead of one is twofold. First, the CDF normal distribution, which we denote by $F_{h,n}(x)$, is a continuous function. Since our objective is to apply the equation 12.5 to determine the number of additional output sessions, a continuous function is well suited to small values of the reference blocking ratio β. On the other hand, the accuracy of the empirical distribution function, denote by $F_{h,e}(x)$ depends on the number of samples, in this case n_w, increasing only in discrete $1/n_w$ steps. Therefore, values of β smaller than $1/(2n_w)$ can only be matched by the maximum observed sample. For example, for a window size of $n_w = 360$ samples, the CDF step size is $2.7 \cdot 10^{-3}$.

Second, it is obvious the normal distribution, despite its continuity properties, does not represent the stochastic error in certain circumstances. In these situations, it becomes imperative to look for an alternative. By considering the observed discontinuity in the measured CDF for morning user activity, we conclude that an empirical approach is better suited than a well-know continuous distribution.

We discriminate between which distribution to use by using a goodness-of-fit test for the normal distribution. This test is applied in real-time over the sampled stochastic error. If the test is passed, the stochastic algorithm uses the normal distribution to compute the output. Otherwise, it resorts to the empirical distribution. Figure 12.5 illustrates the principle of the algorithm.

12.3.1 Fitting to a Normal Distribution

The normal distribution is characterized by the mean μ and the variance σ^2. We estimate these parameters using the sample mean and variance of the stochastic error, $\bar{e}_h(k)$ and $s^2_{e_h}(k)$. Although we can compute these variables using the expressions from equation 12.6, in the interest of efficiency, we use the binomial theorem to expand each parameter into simpler sums. This approach has the advantage of an execution in constant time, rather than proportional with the size of the window.

12.3.1.1 The Distribution Parameters

To compute the sample mean and variance, we compute the sums $S_0(k)$ and $S_1(k)$, where:

$$S_q(k) = \sum_{i=k-n_w+1}^{k} (e_h(i))^q \,. \tag{12.7}$$

By using this method, the sums at sample $k+1$ can be expressed in a recursive manner as:

$$S_q(k+1) = S_q(k) + e_h^q(k+1) - e_h^q(k-n_w+1). \tag{12.8}$$

The sample mean becomes:

$$\bar{e}_h(k) = \frac{S_1(k)}{S_0(k)}, \tag{12.9}$$

while the unbiased sample variance can be expressed as:

$$s_{e_h}^2(k) = \frac{S_0(k)S_2(k) - S_1^2(k)}{S_0(k)\left(S_0(k)-1\right)}. \tag{12.10}$$

Knowing the distribution parameters, the CDF becomes:

$$F_{\mathcal{N}}(x,k) = \frac{1}{2} + \frac{1}{2}\mathrm{erf}\left(\frac{x-\bar{e}_h(k)}{s_{e_h}(k)\sqrt{2}}\right), \tag{12.11}$$

and the inverse (or quantile) is:

$$F_{\mathcal{N}}^{-1}(y,k) = \bar{e}_h(k) + s_{e_h}(k)\sqrt{2}\mathrm{erf}^{-1}(2y-1). \tag{12.12}$$

12.3.1.2 The Normality Test

Testing the normality of empirical data is a problem that has received a lot of interest. Besides historical curiosities, such as the KS statistic, the Shapiro-Wilk test (Shapiro & Wilk, 1965) or the D'Agostino-Pearson omnibus test (D'Agostino & Pearson, 1973) are among the most powerful goodness-of-fit tools. The advantage of the omnibus normality test is that it detects deviations from normality in both skewness and kurtosis. Therefore, we apply the omnibus test statistic to the windowed stochastic error, as recommended by D'Agostino et al. in one of their more recent publications (D'Agostino et al., 1990).

The omnibus test computes a statistic metric, denoted by K^2 (for this reason, the test is also called the D'Agostino's K^2 test). If the null hypothesis of normality is true, the K^2 statistic will have a χ^2 distribution with 2 degrees of freedom. By computing the significance level (or p-value) that K^2 is part of a χ^2 distribution, we can determine whether we can reject the null hypothesis. We have:

$$p = 1 - F_{\chi^2}(K^2|2), \tag{12.13}$$

where:

$$F_{\chi^2}(x|v) = \int_0^x \frac{u^{(v-2)/2}\mathrm{e}^{u/2}}{2^{v/2}\Gamma(v/2)}\mathrm{d}u \tag{12.14}$$

is the cumulative distribution function of the χ^2 distribution with v degrees of freedom.

Therefore, we obtain:

$$p = 1 - \int_0^{K^2} \frac{\mathrm{e}^{-u/2}}{2\Gamma(1)}\mathrm{d}u = 1 - \frac{1}{2}\int_0^{K^2} \mathrm{e}^{-u/2}\mathrm{d}u = e^{-K^2/2}. \tag{12.15}$$

If the p-value is smaller than an acceptable significance level (e.g. 0.05), then the null hypothesis that the data is normally distributed can be rejected. Otherwise, although not certain based on the K^2 statistic alone, we accept the stochastic error samples as normally distributed.

In order to apply the K^2 omnibus test, we need to compute the skewness and kurtosis of the sampled data. We apply a method similar to the computation of mean and variance, by computing the third and fourth power sums S_3 and S_4, in addition to S_0, S_1, and S_2. Hence, the skewness and the kurtosis become:

$$\mathrm{Skew}_{e_h}(k) = \frac{S_0^2(k)S_3(k) - 3S_0(k)S_1(k)S_2(k) + 2S_1^3(k)}{S_0^3(k)s_{e_h}^{3/2}(k)}, \tag{12.16}$$

and:

$$\mathrm{Kurt}_{e_h}(k) = \frac{S_0^3(k)S_4(k) - 4S_0^2(k)S_1(k)S_3(k) + 6S_0(k)S_1^2(k)S_2(k) - 3S_1^4(k)}{S_0^4(k)s_{e_h}^2(k)}, \tag{12.17}$$

respectively.

If the stochastic error is fitted successfully to the normal distribution, knowing the sample mean $\bar{e}_h(k)$ and variance $s_{e_h}^2(k)$, we can compute the number of additional upload sessions, by using the inverse of the CDF of the normal distribution. Hence, we obtain:

$$w_h(k) = F_{\mathcal{N}}^{-1}(1-\beta, k) = \bar{e}_h(k) + s_{e_h}(k)\sqrt{2}\mathrm{erf}^{-1}(1-2\beta). \tag{12.18}$$

12.3.2 Fitting to an Empirical Distribution

The empirical cumulative distribution function (E-CDF) accommodates the situations where the stochastic error samples are not normally distributed, that is the p-value of the omnibus normality test is not greater than the desired significance level, in our case, 0.05. An E-CDF can be determined for any data, and the procedure is straightforward: the sample data, say $x_1 .. x_n$, is sorted in increasing order, such that:

$$x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(n)}, \tag{12.19}$$

where we denoted by $(\cdot)$ the sorting permutation.

The E-CDF is a discontinuous step function, increasing with $1/n$ at each sample. Therefore, we have:

$$F_{\mathrm{ecdf}}(x) = \begin{cases} 0 & x < x_{(1)} \\ 1/n & x_{(1)} \leq x < x_{(2)} \\ 2/n & x_{(2)} \leq x < x_{(3)} \\ \vdots & \\ (n-1)/n & x_{(n-1)} \leq x < x_{(n)} \\ 1 & x_{(n)} < x. \end{cases} \tag{12.20}$$

The E-CDF is also bijective, hence it has an inverse:

$$F_{\mathrm{ecdf}}^{-1}(y) = \begin{cases} -\infty & y < 1/n \\ x_{(\lfloor ny \rfloor)} & 1/n \leq y \leq 1. \end{cases} \tag{12.21}$$

From an implementation perspective, the complexity of using the empirical distribution function raised additional warnings. While it is reasonable to assume that the fitting of the normal distribution can be accomplished in near-constant time $O(1)$ using the power-sums, solving the inverse E-CDF involves two steps.

First, we must sort the stochastic error samples. Since we can optimize the algorithm by storing the previous samples already in a sorted order, only the newest samples have to inserted, an operation usually performed in logarithmic amortized time with the window size $O(\log n_w)$. Second, finding the corresponding element for a given probability is also done in logarithmic time $O(\log n_w)$. Hence, the overall performance of the empirical distribution function depends on the logarithm-squared of the sliding window $O(\log^2 n_w)$. Therefore, we must be careful on whether using the empirical distribution is justified, and whether the execution time is reasonable, considering this findings. In the next sections, we address these issues.

12.3.3 The Stochastic Algorithm: Normal or Empirical?

The stochastic control algorithm uses the normality omnibus test to select whether to use the normal or the empirical cumulative distribution function. Once selected, the algorithm calculates the additional number of upload sessions. Our implementation is done in the C++ programming language and optimized using squared sums for calculating the mean, variance, skewness and kurtosis. However, in the interest of brevity, the algorithm 12.1 summarizes the equivalent steps in Matlab.

Code sample 12.1: The stochastic control algorithm.

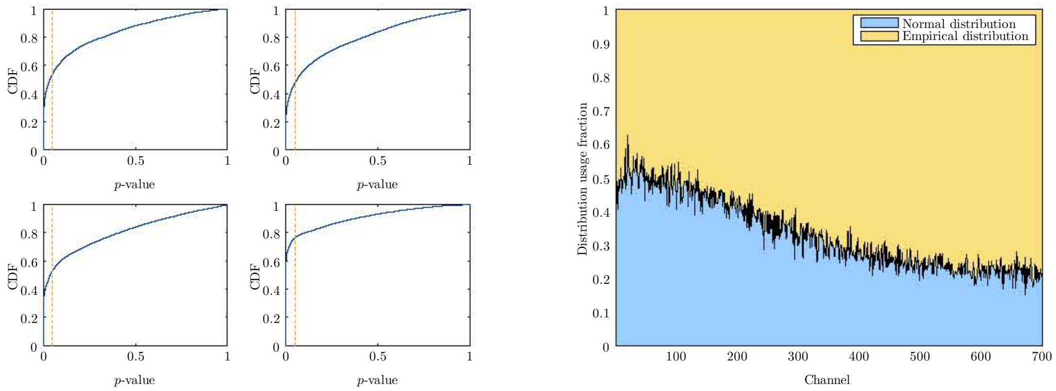
```
% Inputs:
%   k - the sampling moment
%   n_w - the window size
%   e_h - the stochastic error
%   b - the reference blocking ratio
% Input-Output:
%   w_h - the stochastic output
function w_h = StochasticControlAlgorithm(k, n_w, e_h, b, w_h)
    x = e_h(k-n_w+1:k); % Select the k-n_w+1 .. k error samples
    if chi2gof(x) % Execute the omnibus normality test
        % The null hypothesis can be rejected: use empirical distribution
        [f y] = ecdf(x); % Compute the empirical CDF
        w_h(k) = y(find(f>=1-b,1)); % Compute the inverse empirical CDF at 1-b
    else
        % The null hypothesis cannot be rejected: use normal distribution
        mu = mean(x); % Compute the sample mean
        sigma = std(x); % Compute the sample standard deviation
        w_h(k) = norminv(1-b,mu,sigma); % Compute the inverse normal CDF at 1-b
    end
end
```

Finally, we determine whether the usage of the two distribution functions is justified, given the nature of the user activity, and therefore of the stochastic error. Toward this end, we analyze the distribution of the p-value resulting from the omnibus test, and the fraction of time each distribution function is used.

The figure 12.6(a) shows the CDF of the p-value for the four TV channels we commonly illustrated (1, 10, 100, and 500). By emphasizing the 0.05 significance level, we observe that the normality test fails in approximative 50 % of time for popular TV channels, and up to 75 % of the time for unpopular channels. The same results is extended for all TV channels (1 to 700) in the figure 12.6(b). The data suggests that using both algorithms was an appropriate decision, given that the normal distribution alone is not sufficient to describe the stochastic error, especially for unpopular TV channels.

12.4 Performance Evaluation

We implement the stochastic control algorithm to determine its performance relative to the goal of improving the blocking ratio b while offering an adequate utilization ρ of the upload



(a) The distribution of the p-value for four TV channels. If the p-value is smaller than the significance level (set at 0.05, in orange), the null hypothesis is rejected.

(b) The fraction of time the stochastic algorithm uses the normal or empirical distribution for each TV channel.

Figure 12.6: The selection of the normal or the empirical distribution using the D'Agostino-Pearson omnibus normality test.

resources. With our implementation, we simulate an IPTV deployment with 10 000 and 100 000 active subscribers during a period of one week and using the synthetic workload model. In addition, we test the implementation performance by measuring the impact of the algorithm parameters on the execution time. As we explained before, this is particularly important in the case of the empirical distribution function that has a logarithmic-squared complexity with the size of the sliding window.

12.4.1 Impact of the Distribution Function

We use the synthetic user activity from two deployment scenarios of 10 000 and 100 000 active TV subscribers, respectively. For each scenario, we determine the utilization ρ and the blocking ratio of connecting users b when the stochastic control algorithm uses the following distribution function:

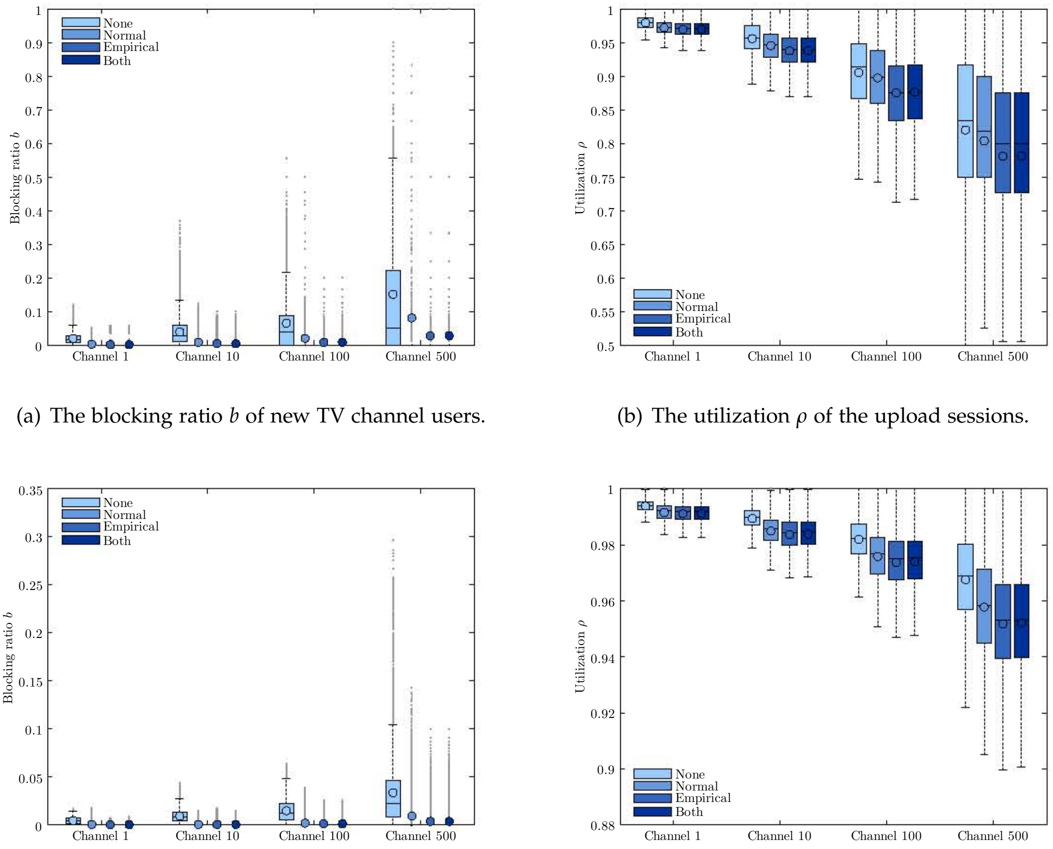
- *none*, that is no stochastic algorithm is applied, to compare with the performance of the fast control algorithm;
- *normal*, exclusively, regardless of the fitting test;
- *empirical*, exclusively, regardless of the fitting test, and;
- *both*, depending on the result of the omnibus normality test at every sampling moment.

The reference blocking ratio for the stochastic algorithm is set to the constant value $\beta = 10^{-3}$.

12.4.1.1 Blocking Ratio

The figures 12.7(a) and 12.7(c) show the blocking ratio b of the upload sessions for the two scenarios we considered, with 10 000 and 100 000 subscribers. These results demonstrate that the stochastic algorithm has a benefic effect, and that the blocking ratio is drastically reduced toward the desired objective $\beta = 10^{-3}$, when using both algorithms and especially for popular TV channels. However, we can also see that the objective is difficult to achieve, especially for unpopular TV channels, where the actual blocking ratio is still one order of magnitude larger than the desired one.

By analyzing the performance across different distribution functions, we note that the normal distribution alone offers a good result only for popular channels, where the difference between



(a) The blocking ratio b of new TV channel users.

(b) The utilization ρ of the upload sessions.

(c) The blocking ratio b of new TV channel users.

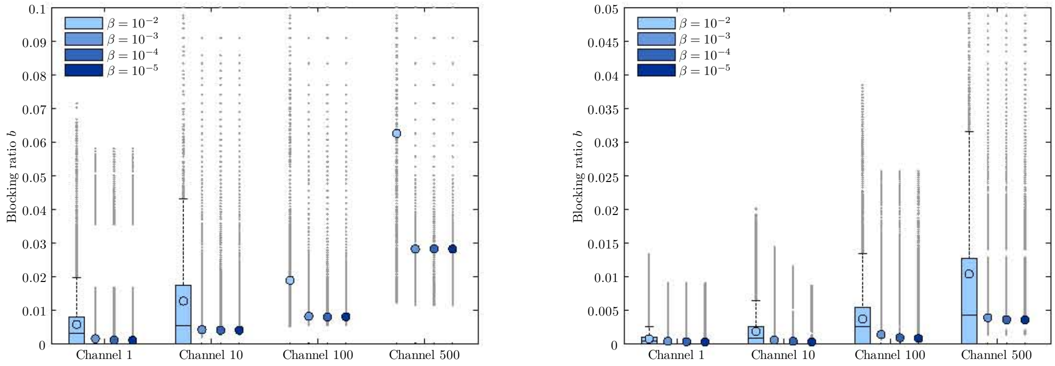
(d) The utilization ρ of the upload sessions.

Figure 12.7: Performance evaluation of the stochastic algorithm versus the algorithm distribution function, in a system with 10 000 active subscribers (top) and 100 000 active subscribers (bottom).

using the normal or the empirical CDF is negligible. For unpopular channels, where the normal CDF characterized poorly the stochastic error, the benefit is much smaller. It is for these situations, where the use of the empirical CDF becomes important. For example, as we can see in the figure 12.7(a) for the channel 500, the ratio between using the normal or the empirical is approximately three times ($\bar{b}_{\text{normal}}/\bar{b}_{\text{empirical}} \approxeq 2.86$).

When used together, the two distribution functions offer the best of both worlds. The blocking ratio is kept at approximately the minimum value between the two, indicating that the normality test correctly discriminates the situations where the normal distribution can be applied (e.g. for channel 500 we have $\bar{b}_{\text{empirical}} \approxeq \bar{b}_{\text{both}} \approxeq 2.81 \cdot 10^{-2}$). However, the disadvantage of the empirical distribution function is that it does not characterize the statistics of a physical phenomenon, in this case users not finding an available upload session when connecting to a TV channel. The E-CDF only illustrates the distribution of a known population, in this case the past history of each channel, being far less accurate in predicting the future evolution of the system. In addition, as we mentioned before, being a step function it has a limited resolution.

For these reasons, while the E-CDF improves the performance in relative terms, attaining the desired blocking ratio is difficult to achieve as it proves to be the case of unpopular channels. Popular Tv channels that rely to a lesser extent on the E-CDF, and have a lower blocking ratio to begin with, prove far less vulnerable to the limitations of the E-CDF. In order to solve the conundrum of the unpopular channels, in the chapter 13 we introduce a third control algorithm, designed specifically to address this issue.



(a) The blocking ratio b of new TV channel users.

(b) The blocking ratio b of new TV channel users.

Figure 12.8: The actual blocking ratio b versus the reference blocking ratio β, in a system with 10 000 active subscribers (left) and 100 000 active subscribers (right).

12.4.1.2 Utilization

Despite the positive effect on the blocking ratio, using the stochastic algorithm has a cost: lower sessions utilization. This outcome can be explained intuitively. The stochastic algorithms allocates supplementary upload sessions to counteract the higher blocking ratio, which reduces the blocking ratio, but at the same time increases the probability that some of those sessions remain unused. After all, the algorithm relies on the statistical expectation of a certain number of users tuning into the TV channel. However, the stochastic algorithm not only increases the number of upload sessions, but it does so in a smart way, for instance allocating fewer sessions during mornings depending on the actual user activity.

The figures 12.7(b) and 12.7(d) illustrate the upload sessions utilization for the scenarios we considered. Although the utilization is lower when using the stochastic algorithm, for the reasons we explained previously, we can see immediately that the actual decrease in utilization is much smaller than the relative decrease in blocking ratio. Namely, if we compare the changes in both blocking ratio and utilization for channel 500, which is easier to observe, the blocking difference is more than three times greater than the utilization difference:

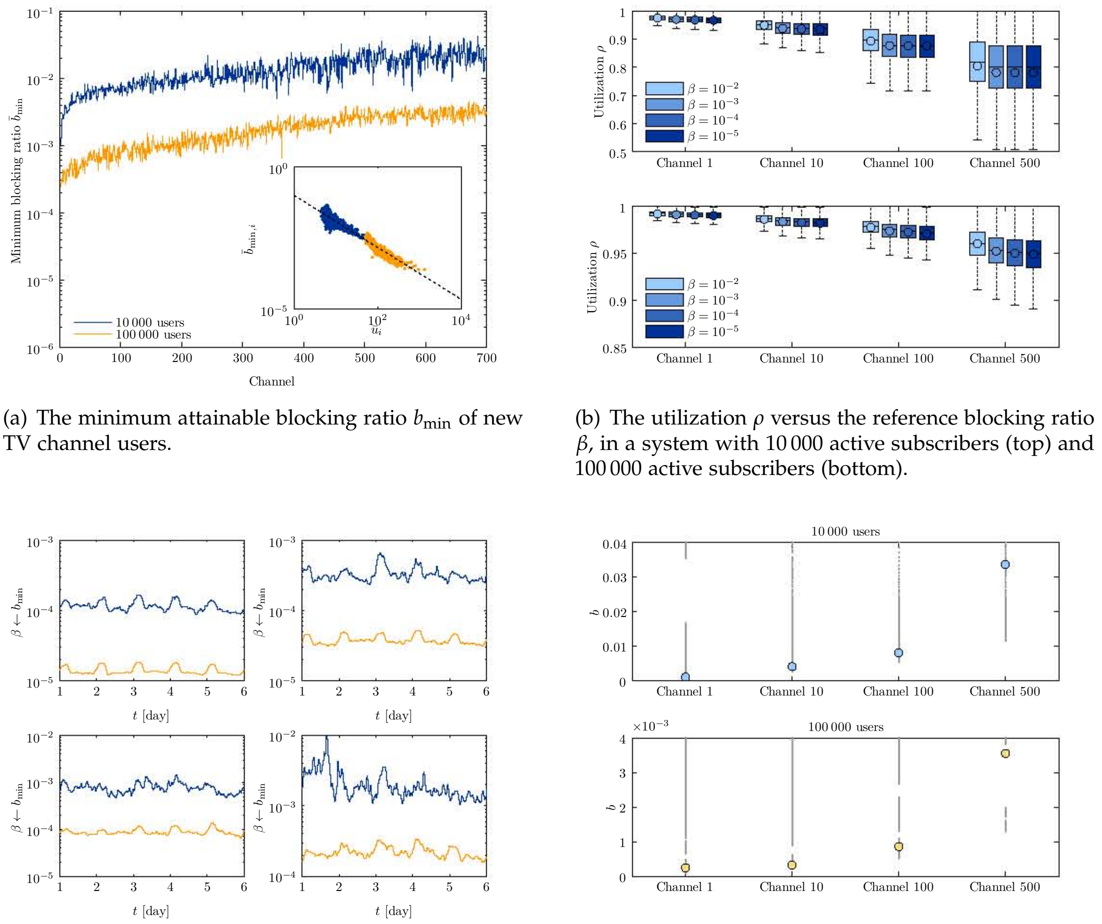
$$\begin{aligned} |\bar{b}_{\text{none}} - \bar{b}_{\text{both}}| &\approxeq 0.12 \\ |\bar{u}_{\text{none}} - \bar{u}_{\text{both}}| &\approxeq 0.04 \end{aligned} \tag{12.22}$$

Essentially, this comparison means that for an unpopular TV channel, the cost in utilization is three times lower than the benefit obtained in blocking ratio. The difference ratio is smaller for popular channels (≈ 2), however the typical performance for those TV channels is usually already close to the optimal values, that is a blocking ratio near the desired β and a utilization near 1.

12.4.2 Impact of the Reference Blocking Ratio

The value of the reference blocking ratio β has a significant impact on the performance of the stochastic algorithm. In particular, due to the uncertainty of the distribution fit, it is impossible to achieve in practice a zero blocking ratio. Therefore, we have to consider the limiting effect of the reference blocking ratio, that is what is the smallest blocking ratio that we obtain in the setting of a given TV channel.

To answer this question, we measure the performance metrics by tuning the blocking ratio



(a) The minimum attainable blocking ratio $b_{\min}$ of new TV channel users.

(b) The utilization ρ versus the reference blocking ratio β, in a system with 10 000 active subscribers (top) and 100 000 active subscribers (bottom).

(c) The value of $b_{\min}$ computed in real-time for channels 1, 10, 100 and 500 (in clockwise order) for deployments of 10 000 (blue) and 100 000 (orange) users.

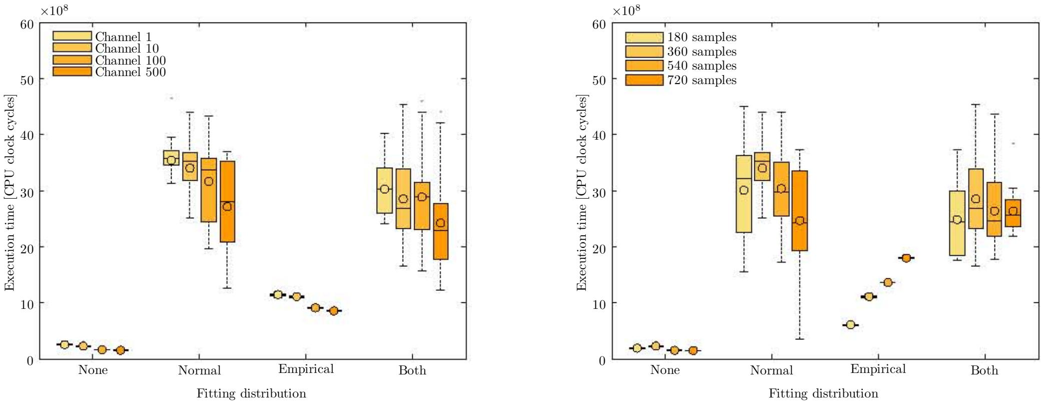
(d) Optimal blocking ratio b for a reference blocking ratio $\beta = b_{\min}$.

Figure 12.9: The minimum attainable blocking ratio $b_{\min}$: characteristics, effects and optimal performance.

within the set of values $\{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$, which we consider a reasonable range. The stochastic algorithm uses both distribution function, depending on the probabilistic nature of the stochastic error, while the IPTV workload represents two deployments of 10 000 and 100 000 active subscribers.

The results, illustrated in the figures 12.8(a)–12.8(b), show that the absolute channel popularity (i.e. the average number of active channels users) plays a decisive role in the performance of the stochastic algorithm. This finding is not surprising, since a larger number of channel subscribers gives a better statistical description of the channel demand. The accuracy of this information is essential for our algorithm, and therefore popular TV channels are easier to predict. For example, the minimum blocking ratio for a deployment of 10 000 users varies between 10^{-3} and $2.8 \cdot 10^{-2}$ for popular and unpopular channels, respectively. If the number of active TV subscriber increases by an order of magnitude, the actual blocking ratio changes to the interval $2.4 \cdot 10^{-4}$ and $3.5 \cdot 10^{-3}$, respectively. Hence, there exists a minimum blocking ratio, denoted by $b_{\min}$, that we can obtain using the fast and stochastic control algorithms alone.

The figure 12.9(a), where we show $b_{\min}$ for all TV channels, makes the correlation to the number of channel users obvious. For instance, the difference of one order of magnitude between the number of subscribers is reflected directly in the mean value of $b_{\min}$ for the same TV channels.



(a) Versus channel popularity.

(b) Versus stochastic error sliding window.

Figure 12.10: Execution performance of the stochastic algorithm.

In the smaller axis of the same figure, we plot the sample mean of b_{min} directly against the mean number of active users on the corresponding TV channel. Using this data, we find an approximate relationship between the two variables, which is linear in logarithmic space, and that can be expressed as:

$$\bar{b}_{\text{min},i} = \mathrm{e}^{\gamma \ln \bar{u}_i + \delta}, \tag{12.23}$$

where $\bar{u}_i$ is the mean number of users watching channel i, and for our user activity we have $\gamma \approxeq -0.91$ and $\delta \approxeq -2.33$.

To understand why the last relationship is important, in the figure 12.9(b), we show the corresponding upload session utilization. The result shows that decreasing the reference β below the minimum capability of the algorithm, b_{min} is usually counter-productive, resulting in a smaller utilization with no significant gain in the blocking ratio. For these reasons, we can use the last equation to determine in real time an appropriate β' that will deliver optimal performance.

12.4.3 Algorithm Benchmarking

The empirical distribution function is an essential component of the stochastic algorithm that is instrumental in lowering the blocking ratio for unpopular TV channels. However, we remember that the implementation complexity with the stochastic error sliding window is typically $O(\log^2 n_w)$. It becomes essential to test whether the algorithm is feasible in practice, given the normal user activity workload, and the typical algorithm parameters.

Toward this end, we submit our C++ implementation to an extensive set of benchmarking tests, in order to measure its real-life performance. In order to determine the contribution of every distribution function, as before, we take each algorithm setting individually, that is *none*, *normal*, *empirical*, and *both*. For benchmarking purposes, we use the optimized version of the code in implementation, as well as compiler and linker settings.

We measure the execution time of the algorithm for each TV channel independently, such that the effects of channel popularity can be quantified easily. The actual testing uses the *Visual Studio Team System Profiler* in instrumentation mode, which inserts measurement probes at C++ function entry and exit points. Although this mode introduces a slight measurement overhead, it is more precise in determining the time spent in each individual function. In addition, the overhead can be precisely determined and excluded from the analysis. Once a benchmarking sessions has completed, the profiling data is exported automatically for further processing.

Code sample 12.2: Benchmarking script using the VSTS profiler.

```
@echo off
echo Profiling started
rem Instrument the executable AS program (only the functions of our algorithm)
VSInstr.exe AppServer.exe /Include:SimIptvControlAdaptive::*
rem Start the VSTS profiler in instrumentation mode
VSPerfCmd /start:trace /output:ProfilingData.vsp
rem Execute the AS program
AppServer.exe config.cfg
rem After completion, stop the profiler
VSPerfCmd /shutdown
rem Generate a CSV report from the data collected by the profiler
VSPerfReport ProfilingData.vsp /Summary:CallTree /Output:ProfilingData.csv
rem Delete the temporary data file
del ProfilingData.vsp
echo Profiling finished
```

To generate a statistical significant result, each measurement is performed 20 times. The figure 12.10(a) shows the algorithm execution time versus each setting of the stochastic algorithm and across the four TV channels. The sliding window size is $n_w = 360$. The execution time is reported in number of CPU cycles for a single channel; this has the advantage of being independent on the processor frequency (although real-life performance may vary slightly across processors of different manufacturers or families). The profiling experiment measures the execution time of the whole control algorithm, that is both *fast* and *stochastic*, over a period of one week.

In absolute terms, executing the algorithm for any distribution function for a single TV channel is extremely fast: running the control algorithm the whole week is accommodated in less than 1 second on a modern class CPU. Since the design requires the stochastic algorithm to run at the end of every sampling period, typical workloads during the sample, such as establishing new sessions, would outweigh the CPU cost of the control algorithm. For these reasons, in this section, we shall focus only on the parametric issues, including the effects of the distribution function and of the sliding window size.

First, we observe that fitting the normal distribution is significantly more CPU intensive (around 3 times) than using the empirical distribution function. This is especially due to the floating point operations of applying the omnibus test and calculating the inverse error function. The execution performance for TV channels with different popularity is similar, since the stochastic algorithm is applied at every sampling moment. Typically, for the execution time, using both distribution functions does not represent a compromise, since the omnibus normality test is required.

Finally, in the figure 12.10(b), we analyze the effect of the sliding window size n_w. We emphasized previously that the implementation of the empirical distribution function, in particular, is susceptible to changes in n_w, by having a logarithmic-square complexity. This fact is visible in the algorithm performance for window sizes between 180 and 720, corresponding to a time interval of 30 minutes up to 2 hours at a sampling rate of 10 seconds. Nevertheless, for the sampling frequency and window sizes we selected, the performance of the empirical version is comparable to the normal one.

12.5 Conclusions

The objective of the *stochastic control algorithm* is to drive the blocking ratio signal $b(k)$ toward a desired reference value β. To accomplish this, the algorithm explores the random nature of the *stochastic error*, that is the difference between the number of users during a sampling period $u(k)$, and the number of upload sessions calculated through fast control $w_f(k)$. The fast control algorithm is designed to keep this difference close to zero. However, the same design cannot guarantee a certain blocking ratio for the subscribers connecting to a TV channel.

By examining in real-time the statistical properties of stochastic error, the stochastic control algorithm determines the number of supplementary upload sessions that are required to meet the blocking ratio target. In essence, it computes the inverse cumulative distribution function of the error signal at the point of the desired blocking ratio.

The statistical distribution of the stochastic error is similar to a normal distribution during high user activity, or highly discontinuous during low user activity. Based on these experimental observations, our algorithm uses fits the error data either to a normal or to an empirical distribution function. The decision on which distribution to use is done according to a normality test.

The experimental validation of the method shows the stochastic algorithm is successful in attaining the blocking ratio goal for popular TV channels. The results are less than optimal in the unpopular case, mainly due to the limited statistical accuracy of the distribution fit. The disadvantage of using the stochastic algorithm is lowering the sessions utilization. Since the reduction in utilization is much smaller compared to the reduction in blocking ratio, we argue this is a reasonable compromise, especially because the compromise level can be controlled through the value of the desired blocking ratio β.

From implementation perspective, the stochastic algorithm is extremely fast for both the normal and the empirical distribution functions. While in theory there are some scalability concerns, they are insignificant in a practical setting.

Chapter 13

The Slow Control Algorithm

13.1 Overview

The *fast* and the *stochastic* control algorithms together offer a good performance of the allocation of upload sessions, in terms of utilization, blocking ratio and implementation complexity. Their only disadvantage is that the blocking ratio performance is generally bounded by the statistical properties of the channel user activity: the larger the number of active channel users, the better the performance. Therefore, while this limit for the blocking ratio is acceptable for popular channel or large deployments (of order 10^{-4} or smaller), it becomes imperative to improve the system performance for unpopular channels.

The user activity on unpopular TV channels has poor statistical properties and therefore it is difficult to predict with the stochastic algorithm. However, since unpopular TV channels experience lesser demand, the number of upload sessions needed to reduce the blocking ratio is many times smaller than for popular channels and it becomes more affordable in terms of upload resources to accommodate a smaller blocking ratio.

Toward this end, in this chapter we introduce a new control algorithm, designed specifically to improve the blocking ratio for unpopular TV channels. Like previously, the algorithm uses a feedback loop to measure the system performance and take a corrective action. In this particular case, the algorithm takes real-time measurements of the channel blocking ratio, $b(k)$, and increases or decreases the number of upload sessions accordingly.

The channel blocking ratio is an approximation of the statistical blocking probability, and measures the fraction of arriving channel users that do not receive an immediate upload session, i.e. are "blocked". In order to be statistical representative, when we introduced the blocking ratio in 11, we explained that is measured over a time frame much larger than a sampling interval, typically between 30 minutes and 1 hour. This means that the changes in the system performance are observed much more slowly than for the fast and stochastic algorithms. For this reason, we call this algorithm a *slow control algorithm*.

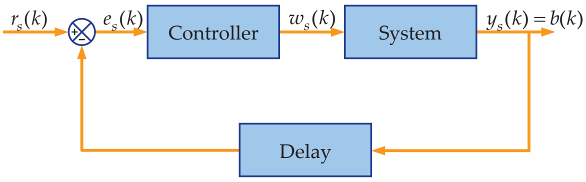


Figure 13.1: The basic design of the slow control algorithm.

13.1.1 Basic Design

The slow algorithm usage of a slow performance metric (the blocking ratio) requires a different design approach. In this case it is impossible to take corrective actions that have immediate effect, and instead we focus on long-term performance enhancement. Unlike the stochastic algorithm, the slow algorithm uses a classic feedback loop design, where the measures output is the blocking ratio:

$$y_s(k) = b(k), \tag{13.1}$$

and the control input, denoted by $w_s(k)$, represents the *additional fraction of upload sessions*, relative to the existing number of users, that have to be allocated for the corresponding TV channel. If the number of upload sessions computed by the fast and stochastic algorithms are $w_f(k)$ and $w_h(k)$, respectively, after including the slow control algorithm, the total number of upload sessions necessary for a TV channel becomes:

$$w(k) = w_f(k) \cdot (1 + w_s(k)) + w_h(k). \tag{13.2}$$

The reason for choosing a relative control input, rather than an absolute one, is that the control output is also relative to the number of existing channel users and upload sessions. Intuitively, reducing a given blocking ratio by a constant amount requires a number of upload sessions than is proportional to the number of users on the channel. However, because the slow algorithm uses a linear control loop, the control input will change only with the output loosing the information on the number of channel users. In order to maintain the system linearity to facilitate a design based on the existing control theory, we keep the control input as a relative one, and we perform the translation to the absolute number of sessions using the multiplication illustrated above.

The figure 13.1 illustrates the basic principle behind the slow control algorithm. The control error $e_s(k)$ is the difference between the desired (or reference) blocking ratio[1] $r_s(k)$ and the measured control output $y_s(k) = b(k)$. A controller function uses the control error signal to compute the system control input $w_s(k)$ that represents the fraction of additional upload sessions. Like in the case of the fast algorithm, the feedback delay represents the latency in the control action, that is a measurement at the sampling moment k will determine a change in the control input at the moment $k + 1$. The dynamic nature of the blocking ratio measurement is included in the transfer characteristic of the whole system and is reflected directly at the control output.

13.1.2 Design Objectives

The slow control algorithm approaches the control problem from a different perspective, when compared with the fast and stochastic algorithm. In particular, the usage of the blocking ratio as a control output means that changes in the control input are not immediately observable. This latency increases the danger of over-reacting in a particular setting leading to system instability.

[1] The reference blocking ratio can be the same as for the stochastic algorithm $r_s(k) = r_h(k)$, unless the stochastic algorithm uses the optimal reference, $r_h(k) = b_{\min}$, that is determined automatically and does not require operator input.

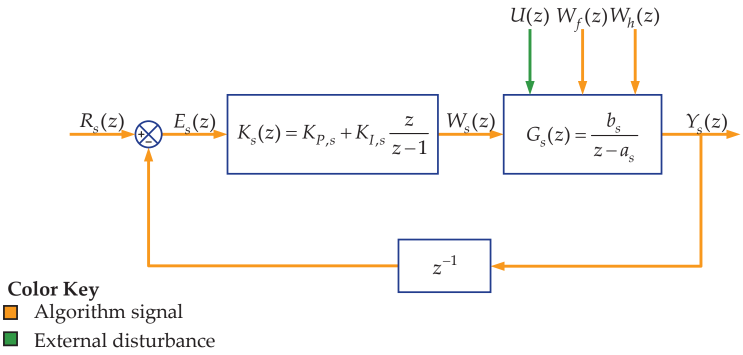


Figure 13.2: The slow control algorithm: the linear control feedback loop.

This problem can be imagined as a situation where the blocking ratio is above the reference value and the control algorithm must increase the number of upload sessions. If the algorithm is too aggressive and it over-provisions the number of sessions, after a measurement delay the blocking ratio drops below the reference. In turn, the small blocking ratio determines an under-provisioning of the number of sessions increasing again the blocking ratio. This process repeats itself, with the control output continuously oscillating around the reference value.

If aggressiveness generates system instability, on the other hand, the algorithm should also not be excessively lazy in responding to disturbances. In the practical setting of a TV channel, the average user demand changes continuously. Therefore, it is important that the slow algorithm responds to the blocking ratio before its value changes significantly, that is requiring a reasonably small setting time.

To summarize, the main design objectives of the slow control algorithm are:

- *stability*, that is the algorithm must not over-react to changes in the control output, leading to oscillations;
- *accuracy*, that is the algorithm must have a sufficiently small settling time, such that the control output follows the reference blocking ratio most of time, and a small or zero maximum overshoot.

13.2 Controller Design

13.2.1 Feedback Loop

The slow algorithm uses a classic linear control feedback loop. Therefore, the algorithm design focuses on the selection of the feedback controller and the computation of its coefficients. Toward this end, in the figure 13.2, we expand the design schematic into the Z-domain and include two important design decisions.

First, we assume that the system, that is the law governing the changes in the blocking ratio due to the user activity and variations in the control input, is a linear one of the first order. Thus, it can be characterized by the linear difference equation:

$$y_s(k+1) = a_s y_s(k) + b_s w_s(k), \tag{13.3}$$

and by the Z transfer function:

$$G_s(z) = \frac{b_s}{z - a_s}. \tag{13.4}$$

Second, we propose a proportional-integral (PI) controller characterized by the coefficient $K_{P,s}$ and $K_{I,s}$. The rationale for choosing a PI controller, rather than a PID one, is the same as for the fast control algorithm: a simpler design and the contribution of the derivative component has limited effectiveness[2]. Hence, the controller transfer function is:

$$K_s(z) = K_{P,s} + K_{I,s}\frac{z}{z-1}, \tag{13.5}$$

where $K_{P,s}$ and $K_{I,s}$ are the proportional and integral controller coefficients.

The reference signal is set at the desired blocking ratio, and every sampling period, the algorithm computes the control input that adjusts the number of upload sessions. Since we design the slow algorithm to complement the stochastic one for unpopular TV channels, depending on the initial system performance, the slow algorithm works both ways and it may increase or decrease the blocking ratio. Therefore, the operator may select between using the slow algorithm for all TV channels (to obtain a uniform blocking ratio and save upload resources), or applying it only for the TV channels where the blocking ratio is reduced (i.e. the control input is positive).

13.2.2 System Model

Unlike the fast control algorithm, where the system transfer function $G_f(z)$ is determined directly, the corresponding input-output law of the slow algorithm is typically a non-linear one. If we approximate our system with an M/M/$w(k)/w(k)$ queue, where the users' arrival and service time is exponentially distributed, and we have $w(k)$ upload sessions at the sampling moment k, the blocking ratio can be approximated by the Erlang-B equation:

$$b(k) \approx \frac{\dfrac{(\lambda/\mu)^{w(k)}}{w(k)!}}{\displaystyle\sum_{i=1}^{w(k)} \frac{(\lambda/\mu)^i}{i!}}, \tag{13.6}$$

where λ is the user mean arrival rate, and $1/\mu$ is the mean channel session duration.

Despite the non-linearity of the blocking ratio, it is possible to assume that the transfer function is linear over a small interval around a given operating point. The assumption makes sense if the control algorithms maintains the output around the reference value. Therefore, we start with the linear transfer function of the first order:

$$G_s(z) = \frac{Y_s(z)}{W_s(z)} = \frac{b_s}{z-a_s}. \tag{13.7}$$

The coefficients a_s and b_s are the pole and the gain of the system, respectively. To identify their values, we resort to a black-box model approach, where we use an artificial input signal, $w_s(k)$, and measure the system response $y_s(k)$. Then, using the least-squares regression method we can estimate the nominator and denominator coefficients (Hellerstein et al., 2004).

We applying this technique to our system in an experimental setting with a triangular test input signal. The amplitude of the test signal around zero is 0.1, which we estimate as an upper (and lower) bound for the fraction of additional upload sessions. To verify the regression method accuracy under different dynamic conditions, we chose a period of the test signal between 30 minutes and 4 hours. We describe in further detail the experimental process in tn appendix C.2.

Table 13.1 illustrates the mean value of the transfer function coefficients, obtained through the least-squares identification method. We evaluate the accuracy of the model, we determine the

[2] In this particular situation, we shall show than the contribution of the proportional component is also limited in terms of improving system performance. However, it plays an important role in system stability

Period of test input $w_s(k)$	$\bar{a}_s$	$\bar{b}_s$	$\bar{R}^2$
30 min	0.9970	$-1.02 \cdot 10^{-5}$	0.9936
60 min	0.9969	$-1.03 \cdot 10^{-5}$	0.9934
120 min	0.9970	$-8.97 \cdot 10^{-5}$	0.9937
240 min	0.9970	$-7.74 \cdot 10^{-5}$	0.9937

Table 13.1: The coefficients of the system transfer function $G_s(z)$.

variability of the modeled data, denoted by R^2, which ranges between 0 and 1. Typically an R^2 greater than 0.8 indicates a good linear model.

13.2.3 The PI Controller

Having determined a linear model for the system, we can compute the coefficients of the PI controller. Similarly with the steps we followed for the fast control algorithm, we use the root-locus method to find the optimal open-loop zero and closed-loop gain.

13.2.3.1 Open-Loop System

The transfer function of the open-loop system (including the feedback delay) is:

$$H_{FF}(z) = K_s(z)G_s(z)H_s(z) = \frac{b_s\left(K_{P,s} + K_{I,s}\right) - b_s K_{P,s}}{z(z-1)(z-a_s)}, \tag{13.8}$$

where $H_s(z) = z^{-1}$ is the one-sample feedback delay.

Hence, the open-loop system has one zero:

$$z_{o,0} = \frac{K_{P,s}}{K_{P,s} + K_{I,s}}, \tag{13.9}$$

and three poles:

$$p_{o,1} = 0, \quad p_{o,2} = a_s, \quad p_{o,3} = 1. \tag{13.10}$$

Given the values we obtained for the system transfer function, we find that the open-loop system is unstable, having one pole on the unit circle.

13.2.3.2 Closed-Loop System

The transfer function of the closed-loop system is:

$$H_R(z) = \frac{K_s(z)G_s(z)}{1 + K_s(z)G_s(z)H_s(z)}, \tag{13.11}$$

and we have the characteristic equation:

$$1 + G_s(z)K_s(z)H_s(z) = 0. \tag{13.12}$$

By replacing with the known quantities, we obtain:

$$1 + \frac{b\left(K_{P,s} + K_{I,s}\right)z - b_s K_{P,s}}{z(z-1)(z-a_s)} = 0, \tag{13.13}$$

or, alternatively:

$$\frac{z^3 - (a_s + 1)z^2 + (a_s + b_s K_{P,s} + b_s K_{I,s})z - bK_{P,s}}{z(z-1)(z-a_s)} = 0. \tag{13.14}$$

The characteristic equation has three roots, resulting in three poles for the closed-loop system $p_{c,1}$, $p_{c,2}$ and $p_{c,3}$. One pole is always real, while the other two can be real or complex. By using the root locus method, we can plot the trajectories of the closed-loop poles as function of closed loop gain, $\kappa \geq 0$. Toward this end, we must rewrite the characteristic equation in the form $1 + \kappa H(z)$, and let:

$$\kappa = b_s \left(K_{P,s} + K_{I,s} \right). \tag{13.15}$$

Under these circumstances, we obtain:

$$1 + \kappa \frac{z - \dfrac{K_{P,s}}{K_{P,s} + K_{I,s}}}{z(z-a_s)(z-1)} = 0, \tag{13.16}$$

or:

$$1 + \kappa \frac{z - z_{o,0}}{z(z-a_s)(z-1)} = 0, \tag{13.17}$$

where the root of the numerator is also the zero of the open-loop system. The characteristic equation becomes:

$$\frac{z^3 - (a_s + 1)z^2 + (a_s + \kappa)z - \kappa z_{o,0}}{z(z-a_s)(z-1)} = 0. \tag{13.18}$$

The root locus depends on the location of the zero with respect to the open-loop poles $p_{o,1}$, $p_{o,2}$ and $p_{o,3}$. Given the known values of these poles, we have four possibilities: (i) $-\infty < z_{o,0} < p_{o,1} = 0$, (ii) $p_{o,1} = 0 < z_{o,0} < p_{o,2} = a_s$, (iii) $p_{o,2} = a_s < z_{o,0} < p_{o,3} = 1$, and (iv) $p_{o,3} = 1 < z_{o,0} < +\infty$. The inequality between the poles and the zero is always strict. Otherwise, when the zero overlaps with an open-loop pole, it reduces the order of the system, a situation that we shall handle separately.

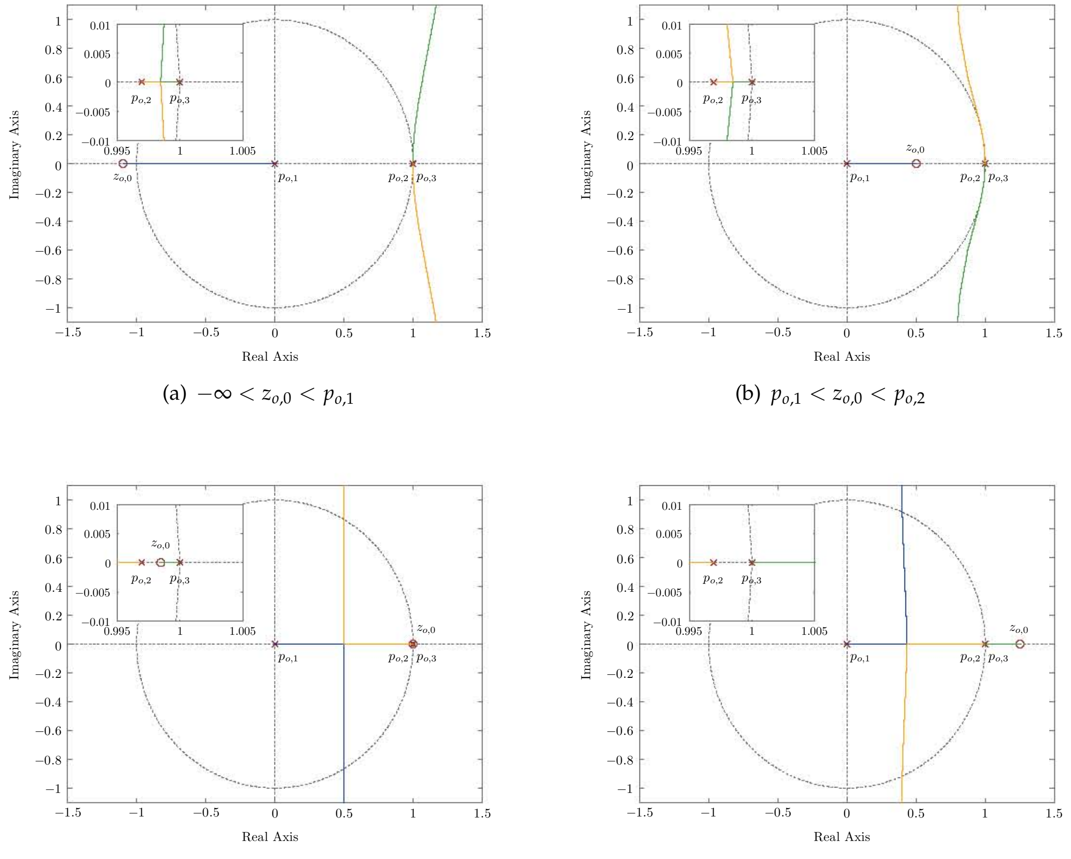
The figures 13.3(a)–13.3(d) illustrate the root loci of the closed-loop characteristic equation, when the open-loop zero is placed according to the aforementioned cases. Each figure shows the trajectory of every closed-loop pole (in a different color), determined by the location of the zero $z_{o,0}$ and for a closed-loop κ between 1 and ∞[3]. By examining the locations of the closed-loop poles, from the perspective of our objectives, i.e. stability, adequate settling and maximum overshoot, we determine a suitable location for the zero.

We remind the reader that the system is stable if all poles are within the unit circle; the settling time is proportional with the magnitude of the largest (or dominant) pole; while the maximum overshot is zero for positive a real dominant pole, and proportional to the pole magnitude for a negative or complex dominant pole.

We assume that the open-loop zero is not equal to any of the open-loop poles. In these circumstances, the close-loop system has three poles, whose loci are illustrated in the figures 13.3(a)–13.3(d). By examining each case, we conclude the following.

- When $-\infty < z_{o,0} < a_s$ (cases (i) and (ii)), the closed-loop system is stable only for a narrow range in the closed-loop gain κ. In addition, the two poles in the right half-plane are real for even a smaller interval for κ. This behavior makes this cases undesirable since a relatively small increase in the loop gain, results in complex poles and, hence, non-zero maximum overshoot, and poles outside the unit circle and an unstable system.

[3] The closed-loop poles start at an open loop pole (for $\kappa = 0$) and tend to either a zero or $\pm\infty$ (for $\kappa = \infty$).



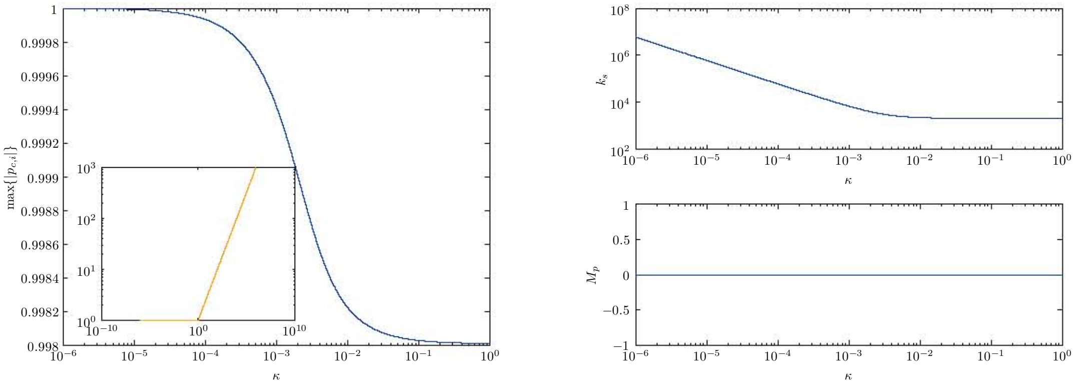
(a) $-\infty < z_{o,0} < p_{o,1}$

(b) $p_{o,1} < z_{o,0} < p_{o,2}$

(c) $p_{o,2} < z_{o,0} < p_{o,3}$

(d) $p_{o,3} < z_{o,0} < +\infty$

Figure 13.3: The root loci of the closed-loop characteristic equation for the third-order system.



(a) The magnitude of the dominant closed-loop pole ($\max\{|p_{c,i}|\}$).

(b) The settling time (k_s) and maximum overshot (M_p).

Figure 13.4: Closed-loop system performance metrics as function of the closed-loop gain κ.

- When $1 < z_{o,0} < +\infty$ (case (iv)), the system is always unstable, as one pole is always outside the unit circle.
- When $a_s < z_{o,0} < 1$ (case (iii)), the real pole (in green) is always inside the unit circle, while the other two are complex and have large magnitudes only for a very large closed loop gain.

We select $a_s < z_{o,0} < 1$ since this situation is the most robust for systems where the closed-loop gain may fluctuate outside the model parameters. In addition, the value of a_s, determined through experimental identification, is close to 1, making the selection of the zero straightforward.

If the closed-loop gain is not too large, all poles are real and the closed-loop system have no overshoot. The settling time is governed by the magnitude of the dominant pole, i.e. $z_{o,0} \leq p_{c,3} < 1$. Because the dominant pole is very close to 1, the system has a long settling time, hence the name of *slow control algorithm*. The minimum settling time is obtained when $\kappa = 0$ and increases with the closed-loop gain. The slow response of the closed-loop is desirable in this situation, as our objective is the improvement of the long term blocking ratio.

If we select the open-loop zero slightly higher than a_s, we can tune the closed-loop gain κ to obtain the desired settling time and avoid overshoot. Considering the typical value of $a_s \approxeq 0.997$, we set $z_{o,0} = 0.999$. Toward this end, in the figure 13.4(a), we analyze the absolute value of the dominant closed-loop pole as function of the closed-loop gain κ. First, we observe that the system *is stable if and only if* $\kappa < 1$. For a larger gain, the complex poles are placed outside the unit circle. Second, with the magnitude of the dominant pole known, we can compute the settling time and maximum overshoot using equations 11.7 and 11.10, respectively. These results, depicted in the figure 13.4(b), show that we can improve the system response time by increasing the overall gain (but carefully such that we do not exceed the stability boundary). The maximum overshoot is always zero since the dominant pole is real and positive.

13.2.3.3 Controller Coefficients

The controller coefficients are computed from the closed-loop gain (κ) and the open-loop zero ($z_{o,0}$). We have:

$$\begin{aligned} K_{P,s} &= -\frac{\kappa z_{o,0}}{b_s}, \\ K_{I,s} &= -\frac{\kappa\left(1 - z_{o,0}\right)}{b_s}. \end{aligned} \tag{13.19}$$

The main challenge in selecting a proper closed-loop is the gain the of the system, b_s. Experimental data shows that b_s fluctuates greatly over several orders of magnitude, depending on the channel activity. While the population median and mean are typically around -10^{-5}, the gain can increase up to -10^{-3} in certain situations. Since the closed-loop gain, κ, is proportional to the system gain, it is dangerous to under-estimate it since the system is stable only for $\kappa < 1$. Hence, we select $b_s \approx -10^{-3}$ at the lower range, such that we are confident in a large margin of safety.

On the other hand, the closed-loop performance, namely the settling time, is relatively constant for κ in the range $10^{-3} .. 1$, and only then increases sharply as κ decreases to zero (see the figure 13.4(b)). Therefore, for a reasonable performance, we can set $\kappa \approx 10^{-3}$, which gives an additional three orders of magnitude margin to the level of instability.

With these selections, we obtain:

$$\begin{aligned} K_{P,s} &\approxeq 1, \\ K_{I,s} &\approxeq 10^{-3}. \end{aligned} \tag{13.20}$$

We remind the reader, that these coefficients can be easily tuned within a large interval around these values with a good margin without affecting the stability or the performance of the closed-loop system. Instead, they are intended as reference values that deliver adequate and robust performance. In the next section, we analyze experimentally and in greater depth how variations of the controller coefficients affect the real IPTV system in terms of blocking ratio and utilization.

13.2.4 Implementation Aspects

Similar to the fast control algorithm, the implementation of the slow control algorithm is extremely simple. The algorithms measures at each samples the total number of channel users arrivals, $n_t(k)$, and the number that are blocked due to insufficient upload sessions, $n_b(k)$. The blocking ratio (or the control output) is determined over a window n_w of the same type and size with the one used for the stochastic error.

The slow error, $e_s(k)$, is computed between the blocking ratio and the desired reference value r_s. Finally, by applying the controller difference equation, the algorithm computes the control input $w_s(k)$, which is the fraction of additional upload sessions, relative to the active channels users, that have to be allocated. The algorithm 13.1 illustrates these steps in Matlab.

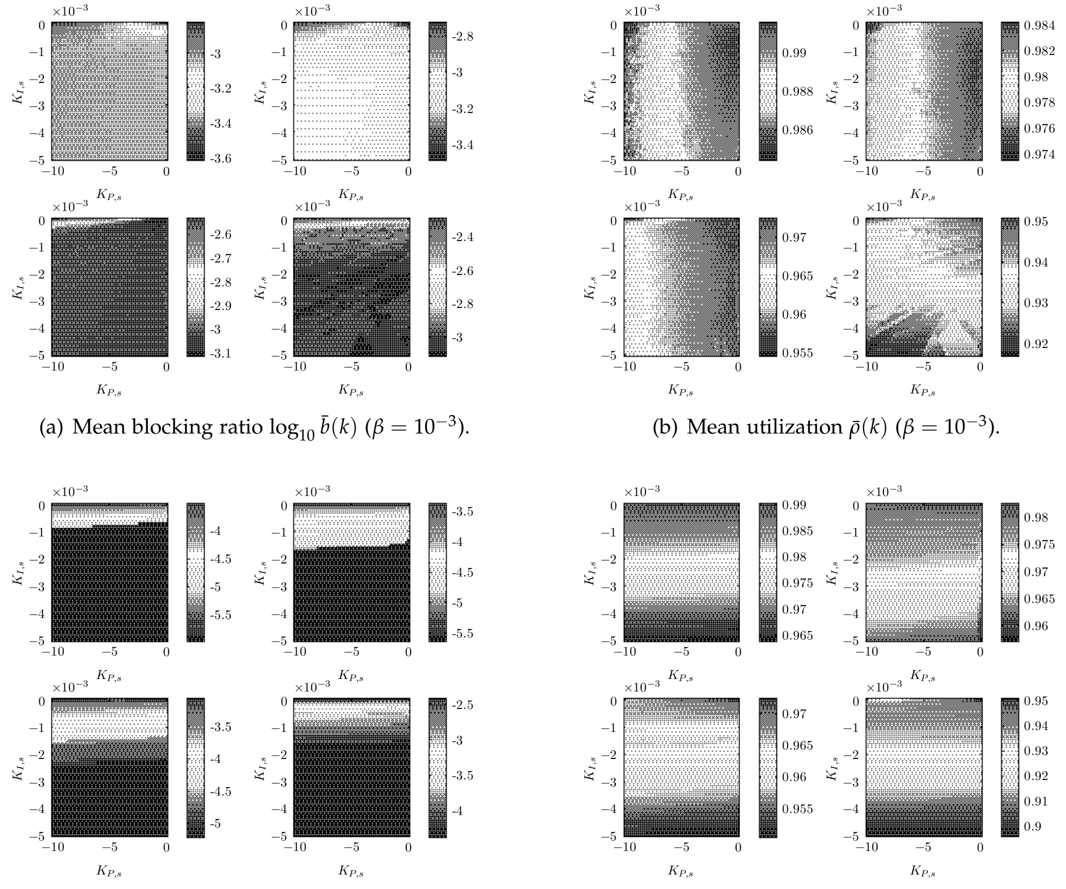
Code sample 13.1: The slow control algorithm.

```matlab
% Inputs:
%   k - the sampling moment
%   n_w - the window size
%   n_t - the total number of user arrivals
%   n_b - the number of blocked user arrivals
%   r_s - the reference blocking ratio
%   K_P - the proportional controller coefficient
%   K_I - the integral controller coefficient
% Input-Output:
%   y_s - the slow control output
%   e_s - the slow control error
%   w_s - the slow control input
function w_s = SlowControlAlgorithm(k, n_w, n_t, n_b, r_s, K_P, K_I,
                                    y_s, e_s, w_s)
    y_s(k) = sum(n_b(k-w+1:k))/sum(n_t(k-w+1:k)); % Compute the blocking ratio
    e_s(k+1) = r_s - y_s(k); % Compute the control error
    w_s(k+1) = w_s(k) + K_P * (e_s(k+1)-e_s(k)) + ...
        K_I * e_s(k+1); % Compute the control input
end
```

13.3 Performance Evaluation

To examine the performance of the slow control algorithm, we simulate an IPTV infrastructure with 100 000 active subscribers during a period of one week using a synthetic user activity. As usual, the performance metrics are the upload session utilization, ρ, and the new user blocking ratio b, on every individual TV channel. In order to keep the results simple, we limit our analysis to the channels 1, 10, 100 and 500, which we believe are representative for the entire channel palette from the popular to the unpopular.

Both the stochastic and the slow algorithms have the same objective: reducing the blocking ratio below a desirable reference value β. However, they achieve the objective through different means. The stochastic algorithm explores the statistical nature of the user activity and works well for popular TV channels a higher β, while the slow algorithm uses a classic linear control loop and is designed for unpopular TV channels and smaller β. Furthermore, the stochastic algorithm determines automatically a reference β for which the algorithm achieves an optimal blocking ratio, without compromising too much on the session utilization, while the slow algorithm always uses a reference blocking ratio selected by the operator.



(a) Mean blocking ratio $\log_{10}\bar{b}(k)$ ($\beta = 10^{-3}$).

(b) Mean utilization $\bar{\rho}(k)$ ($\beta = 10^{-3}$).

(c) Mean blocking ratio $\log_{10}\bar{b}(k)$ ($\beta = 10^{-4}$).

(d) Mean utilization $\bar{\rho}(k)$ ($\beta = 10^{-4}$).

Figure 13.5: The impact of controller parameters, $K_{P,s}$ and $K_{I,s}$, on the performance of the slow algorithm for TV channels 1, 10, 100 and 500 (in clock-wise order from left-top).

Due to the different level of performance between the stochastic and the slow algorithms, in situations where there is a conflict we must decide which algorithm to use. Since only the slow algorithm uses a reference chosen by the operator, we have *two modes of operation*, and the rule is the following:

- In the *priority mode*, the output of the slow algorithm takes precedence over the stochastic one. In other words, we select the number of upload sessions that give a blocking ratio closer to the operator-selected reference.
- In the *non-priority mode*, we select the maximum number of upload sessions between the output of both algorithms. This mode gives the minimum blocking ratio at a compromise with a reduced utilization.

13.3.1 Impact of the Controller Coefficients

Our analysis indicates that we can set the coefficients of the feedback controller, $K_{P,s}$ and $K_{I,s}$, within a large range of values, without significantly affecting the settling time and maximum overshot. Toward this end, we selected a pair of coefficients that offer a good performance and have a significant margin to the limit of stability. To verify the coefficients selection was a good one, we measure the slow control algorithm performance for a set of values for $K_{P,s}$ and $K_{I,s}$. By varying the acceptable closed-loop gain, we obtain:

Mode	β	Channel 1	Channel 10	Channel 100	Channel 500
Priority	10^{-3}	$8.906 \cdot 10^{-4}$	$9.217 \cdot 10^{-4}$	$9.667 \cdot 10^{-4}$	$7.960 \cdot 10^{-4}$
	10^{-4}	0	0	$1.646 \cdot 10^{-5}$	$4.213 \cdot 10^{-5}$
	10^{-5}	0	0	$6.072 \cdot 10^{-6}$	0
	0	0	0	$6.072 \cdot 10^{-6}$	0
Non-priority	10^{-3}	$2.409 \cdot 10^{-4}$	$3.142 \cdot 10^{-4}$	$7.110 \cdot 10^{-4}$	$7.960 \cdot 10^{-4}$
	10^{-4}	0	0	$1.646 \cdot 10^{-5}$	$4.213 \cdot 10^{-6}$
	10^{-5}	0	0	$6.072 \cdot 10^{-6}$	0
	0	0	0	$6.072 \cdot 10^{-6}$	0

Table 13.2: The mean blocking ratio $\bar{b}$.

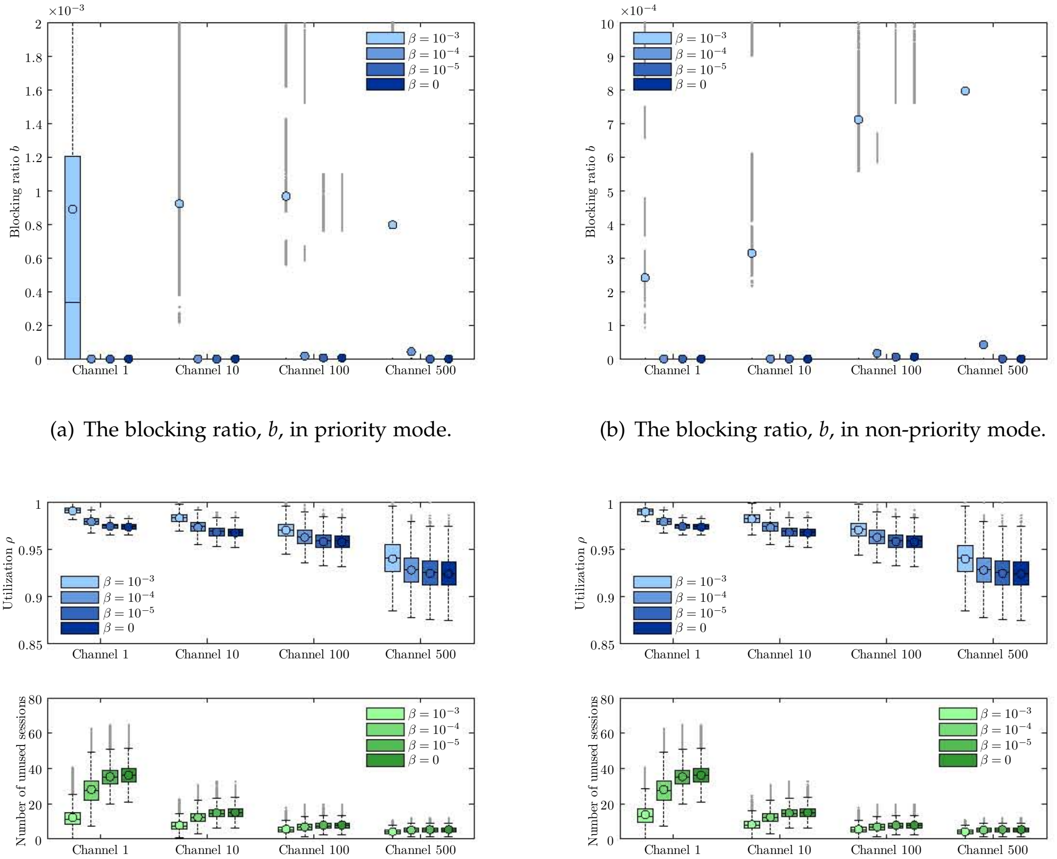
$$\begin{aligned} K_{P,s} &= -10..0, \\ K_{I,s} &= -10^{-2}..0, \end{aligned} \tag{13.21}$$

The figures 13.5(a)–13.5(d) illustrate the mean blocking ratio, $\bar{b}$, and upload sessions utilization, $\bar{\rho}$, for every pair $(K_{P,s}, K_{I,s})$. We repeat the measurements for two different values of the reference blocking ratio, β: the figures 13.5(a), 13.5(b) show the performance metrics for $\beta = 10^{-3}$; the figures 13.5(c), 13.5(d) for $\beta = 10^{-4}$, respectively. The application server is running in priority mode, to emphasize the effect of the pair of coefficients.

Focusing first on the blocking ratio (the figures 13.5(a) and 13.5(c)), in both cases, the result indicates the robustness of the control coefficients, where the blocking ratio (we illustrate the base-10 logarithm for legibility purposed) is kept around or bellow the desired reference for typical coefficient values. Because we take the coefficients all the way to zero, corresponding to no slow control, the contribution of the slow control algorithm becomes obvious especially when the reference, β, is smaller. The importance of the integral component is emphasized as well, and we find that values of 10^{-3} or larger are more effective against the deviations of the blocking ratio from the reference value. This finding is not surprising, since we rely on the integrator to minimize the steady-state error of the blocking ratio that cannot be reduced through the fast and stochastic approaches.

As an unusual situation, when the reference is greater than the blocking ratio obtained with the fast and stochastic algorithms alone (such as $\beta = 10^{-3}$ and for the popular TV channels 1 and 10 in the figure 13.5(a) top), we see the blocking ratio increasing as a result of the slow algorithm running in priority mode.

The utilization, illustrated in the figures 13.5(a) and 13.5(c), complements the results on the blocking ratio, given the effects we already observed. First, the changes in the utilization occur over a very narrow range, even though the blocking ratio is decreased in certain situations by an order of magnitude or more. Second, the utilization is inverse proportional to the control coefficients. This result is not surprising, since as in the case of the stochastic algorithm, the utilization is sensitive to the aggressiveness of the algorithm. In practice, the utilization can be more sensitive to changes in $K_{P,s}$, $K_{I,s}$ or both coefficients, depending on the actual experimental setting (user activity and reference blocking ratio).



(a) The blocking ratio, b, in priority mode.

(b) The blocking ratio, b, in non-priority mode.

(c) The session utilization in priority mode.

(d) The session utilization in non-priority mode.

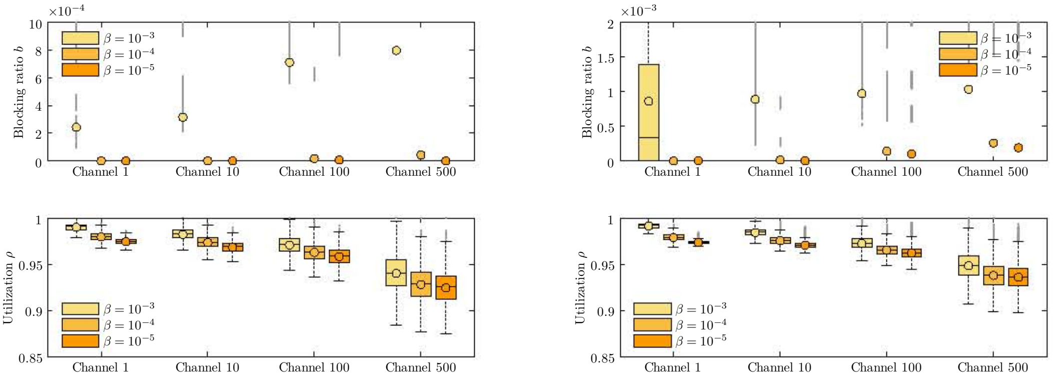
Figure 13.6: The performance of the slow control algorithm, versus the reference blocking ratio, β.

13.3.2 Impact of the Reference Blocking Ratio

Next, we analyze the nominal performance of the slow algorithm by measuring the blocking ratio and sessions utilization against the reference blocking ratio, β. Although in the previous section we showed the flexibility we have for the selection of the controller coefficients, the settings we used in this experiment are the default values from the equation 13.20. The reference blocking ratio is set at 10^{-3}, 10^{-4}, 10^{-5}, and 0 (as a limiting value).

The figures 13.6(a) and 13.6(b) illustrate the blocking ratio for the priority and non-priority modes, respectively. For legibility purposes, the table 13.2 repeats the population mean of the blocking ratio, such that its magnitude is clear. From these results, it becomes obvious that the slow control algorithm is successful at maintaining the blocking ratio around or below the reference value set by the operator. In addition, its performance is less sensitive to the channel popularity, a fact emphasized when the algorithm is used in the priority mode. In non-priority mode, when the slow algorithm does not interfere with system performance better than the reference, the blocking ratio is even smaller for popular TV channels.

We remark however that, as in the case of any real-life system, there may be extremely rare situations that our algorithm cannot compensate. For example, this is true in the case of the TV channel 100, where the blocking ratio cannot be lowered below $6 \cdot 10^{-6}$, even when the reference is set to zero. However, we believe such a result is completely acceptable, given the random component of the user activity, our limited experimental setting, and the fixed parameters of our algorithm that may not accommodate any extreme situation.



(a) With the stochastic control algorithm in *non-priority* mode.

(b) Without the stochastic algorithm.

Figure 13.7: The performance benefit of the stochastic algorithm.

The upload session utilization varies according to the channel popularity, such that unpopular channels determine a poorer allocation of resources. However, we have to remember that unpopular channels also have fewer users, and therefore the absolute number of sessions that go unused is similar to the popular channels. To put this in perspective, the figures 13.6(d) and 13.6(c) show both the utilization, u, and the number of unused upload sessions[4]. These results demonstrate the effectiveness of the combined algorithms to maintain a near-zero blocking ratio for new arrivals while a providing a near-full utilization of the allocated resources. For the unpopular channels, where the apparent utilization is much lower, we must consider the absolute session usage that shows that only a small number of sessions (typically less than 10) are unused.

13.3.3 Impact of the Stochastic Algorithm

The slow control algorithm complements the operation of the stochastic algorithm, and targets in particular unpopular TV channels with poor statistical properties. Given that the two algorithms solve the same problem, it is essential to determine whether the redundance of two parallel algorithms is justified. Toward this end, we disable the contribution of the stochastic algorithm, evaluate the system performance when using only slow control, and compare it to the stochastic-slow algorithms together in non-priority mode[5].

The figures 13.7(a) and 13.7(b) illustrate the results of our comparison, and we conclude that the slow control algorithm alone works well for popular TV channels, but performs sub-optimally for the unpopular ones, and therefore defeating the purpose of its design. The root cause of this behavior lies in the changed dynamics of the blocking ratio (when the stochastic algorithm is not used), and its influence on the slow controller. The stochastic algorithm provides a performance boost to the whole system, by smoothing out the blocking ratio, which is then easier to control. The slow algorithm has a slow response compared to the stochastic version (typically by an order of magnitude), and its current design works well if the large transients of the blocking ratio are compensated with a different method. Finally, using both algorithms in non-priority mode has a secondary benefit for the popular channels, where the blocking ratio can be reduced fourth-fold with negligible impact on the utilization.

[4] During the experimental determination of this result, we assume that a new upload session is allocated immediately after a change of the corresponding variable at the application server. In practice, the number of unused sessions is slightly lower, due to session establishment delay.

[5] The priority mode is of little interest for this comparison because in priority mode the slow algorithm takes precedence.

13.4 Conclusions

The *slow control algorithm* complements the stochastic control in obtaining a blocking ratio, $b(k)$, for arriving channel users, which is close to the reference value, β, selected by the operator. Because the stochastic control cannot correct a steady-state error of the blocking ratio that lies outside its design envelope, the slow control uses a feedback loop to actively measure the instantaneous channel blocking ratio and allocate supplementary upload sessions when required.

In order to determine the amount of correction needed, we assume the system dynamics, that is the change in the blocking ratio (output) as function of the supplementary sessions normalized by the number of users (input), is linear and we find its corresponding transfer function using experimental identification based on least-squares linear regression.

To compute the control input during each sampling period, we use a proportional-integral (PI) controller. We determine its coefficients by relying on a root-locus approach that considers common issues to feedback control systems, such as stability, low response time (settling time) and minimum overshoot. In addition, by selecting these coefficients with a large margin of safety, we ensure the system is sufficiently robust to unpredicted deviations from the normal behavior.

The performance analysis shows that the slow algorithm is instrumental in achieving the desired blocking ratio. Combined with the fast and stochastic control methods, it performs extremely well under the adverse conditions of TV channels of varying popularity, and user activity periods of different intensity. In this manner, the application server easily accommodates the viewers demand with a minimum allocation of extra bandwidth resources.

Since both the stochastic and the slow algorithms perform a similar function, we verify if both algorithms are required in a practical situation. Our analysis shows that the stochastic algorithm has a welcomed smoothing effect on the blocking ratio, even for unpopular channels, making the slow control system easier to design. When working together, both algorithms can achieve very small blocking ratios, which otherwise would not be possible with any algorithm by itself.

Chapter 14

The Peer Coordination Algorithm

14.1 Overview

In the previous chapters, we introduced the session coordination algorithm of the P2PTV-AS, which estimates the number of upload sessions, for each TV channel, necessary to accommodate the viewer demand, and implement the fast signaling enhancement. In contrast, the *peer coordination algorithm*, or PCA, focuses on allocating the UE peer bandwidth resources to these sessions, while addressing the objectives of peer churn enhancement. In a nutshell, the role of the peer coordination algorithm is to satisfy the uploading requests, which involves the following operations.

- Tracking the available UE peers, their network resources, and their downloading/uploading sessions.
- Assigning the peer bandwidth across the existing streams, according to the viewing demand.
- Selecting a suitable peer that can service an outstanding session request, where, depending on the situation, the session may be either active or inactive.

The main challenge in the implementation of the P2PTV service is the uploading churn generated by viewers changing their current TV channel. Unlike the typical peer churn caused by viewers turning off their user equipment, the channel changes represent a much more common activity and may significantly affect performance. To counteract the damaging effect of the channel-change churn, Wu et al. are the first authors to propose a P2PTV system called view-upload decoupling, or VUD, where the peer upload is independent from the current active TV channel (D. Wu, Liang, et al., 2009). This approach generates a more stable streaming overlay, only affected by the UE peers arriving and leaving the system. The drawback is the increased bandwidth usage, since peers must download both the active channel and the uploading streams.

Understanding the inflexibility of the VUD algorithm, related to the separation of view and upload, Wang et al. suggest a more general approach where peers also contribute on their current channel, with the objective of enhancing the utilization of their resources (M. Wang et al., 2010).

They distinguish between three different modes: a naïve bandwidth approach (NBA), where peers upload only their current channel and allocate bandwidth proportional to the streaming rates; a passive channel-aware approach (PCAA), where, like NBA, peers upload only their current channel, but optimize the bandwidth allocation; and, finally, an active channel-aware approach (ACAA) with peers upload multiple channels, subject to the available bandwidth[1].

Unfortunately, the problem behind ACAA is that the designers focus on the bandwidth satisfaction as their main performance metric, considering the channel-change delay a secondary concern. For this reason, unlike VUD, ACAA peers are designed to maximize the bandwidth utilization, rather than preventing streaming interruptions due to churn.

14.2 Streaming Mechanisms

Compared to the VUD algorithm, the setting of our P2PTV service presents a few key differences. First, the IMS platform requires the reservation of networking resources, and therefore a fixed participation of all UE peers in an uploading channel may unnecessarily waste the network bandwidth. Second, the session coordination algorithm determines the number of necessary upload sessions, which enables us to be more flexible toward assigning peers to the uploading streams.

14.2.1 Bandwidth Allocation

As shown in the figure 4.16, on the downloading side, the UE peers maintain two types of sessions: *primary* mainly to download the viewing channel, whose set is denoted by $\mathcal{P}$, and *secondary* for the uploading streams of one or more channels, included in the set $\mathcal{S}$. On the uploading side, the sessions may be either active or inactive, depending on the request received from the P2PTV-AS.

We begin with introducing the following notations.

Notation		Description
b_s		The stream bandwidth
b_d	$b_d(i)$	The downlink bandwidth (for peer i)
b_u	$b_u(i)$	The uplink bandwidth (for peer i)
n_d	$n_d(i)$	The number of downloading streams (for peer i)
n_i	$n_i(i)$	The number of uploading inactive streams (for peer i)
n_p	$n_p(i)$	The number of downloading primary streams (for peer i)
n_s	$n_s(i)$	The number of downloading secondary streams (for peer i)
n_u	$n_u(i)$	The number of uploading streams (for peer i)

Table 14.1: The notations used for the peer coordination algorithm.

For a selected peer with uplink bandwidth b_u and downlink bandwidth b_d, the maximum number of supported upload and download streams, respectively, is:

$$\begin{aligned} n_{u,max} &= \left\lfloor \frac{b_u}{b_s} \right\rfloor, \\ n_{d,max} &= \left\lfloor \frac{b_d}{b_s} \right\rfloor. \end{aligned} \tag{14.1}$$

[1] The original publication used the acronyms PCA and ACA for the passive and the active channel-aware approaches. In this chapter, we changed their acronyms to PCAA and ACAA, respectively, to prevent any confusion with the acronym for the peer coordination algorithm.

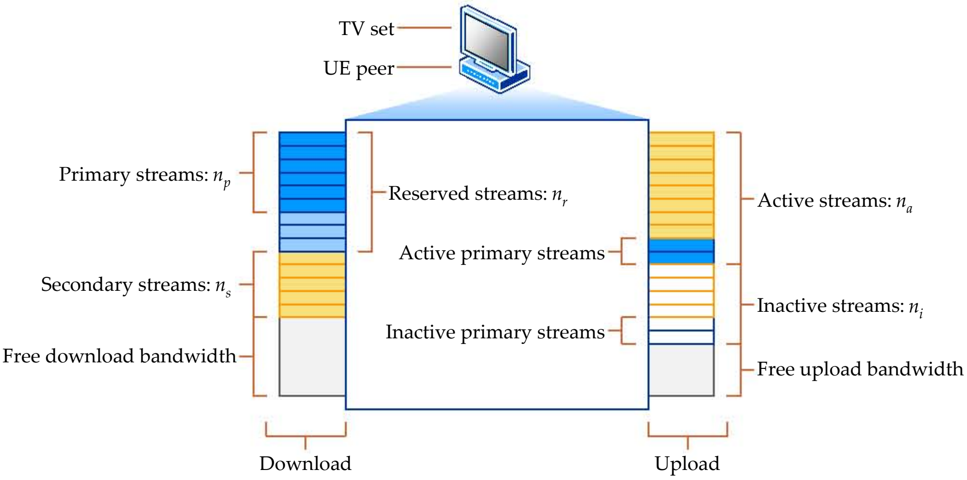


Figure 14.1: The UE peer bandwidth allocation.

The actual number of streams may be lower, $n_u \leq n_{u,max}$ and $n_d \leq n_{d,max}$, depending on the selected channel and the peer participation.

All peers reserve n_r downloading streams for the primary TV channel, a configuration parameter that is selected by the service provider. We note that the reserved streams represent a setting internal to the UE peer and the peer state at the PCA, and do not imply a multimedia session or bandwidth committed within the transport network. Its purpose is to ensure sufficient free bandwidth when changing between TV channels.

The value of n_r may be insufficient for the bit-rate requirements of all TV channels. This is especially true when the channel bandwidth is not uniform, and the provider wishes the maximization of the available resources at the expense of a higher setup delay. For example, an high-definition (HDTV) channel may use up to five times as many streams, compared with standard-definition (SDTV). However, in practice there may be far fewer HDTV channels, and therefore reserving bandwidth for a large number of primary streams will be prohibitively expensive.

The figure 14.1 illustrates the bandwidth allocation of a typical UE peer, where:

$$\begin{aligned} b_{d,c} &= \left(n_p + n_s\right) b_s, \\ b_{d,r} &= \max\left\{\left(n_r - n_p\right), 0\right\} b_s, \\ b_{d,f} &= b_d - \left(n_r + n_s\right) b_s, \end{aligned} \tag{14.2}$$

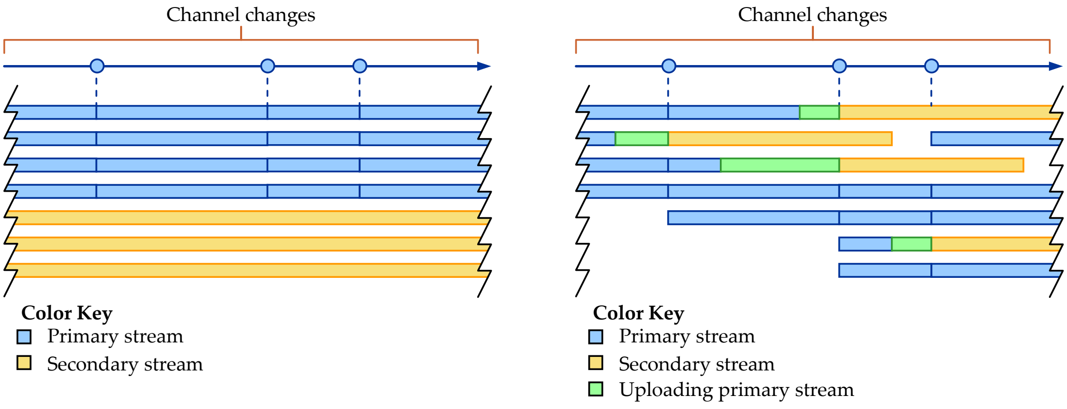
is the committed, reserved but not committed, and free download bandwidth, respectively, and:

$$\begin{aligned} b_{u,c} &= \left(n_a + n_i\right) b_s, \\ b_{u,f} &= b_u - \left(n_a + n_i\right) b_s, \end{aligned} \tag{14.3}$$

is the committed and free upload bandwidth, respectively.

14.2.2 Decentralized Coordination

Similar to both VUD and ACAA, the P2PTV UE peers employ a loose decoupling between viewing and uploading. In principle, the active TV channel viewed by the user is decoded from the primary streams, while a variable number of secondary streams are downloaded and uploaded to



(a) View-upload decoupling.

(b) Peer coordination algorithm.

Figure 14.2: Comparison between the allocation of view (primary) and upload (secondary) peer bandwidth.

neighboring peers, subject to sufficient bandwidth. However, unlike VUD, our algorithm allows peers to upload both primary and secondary streams, conditioned that the primary streams can become secondary streams after the peer moves to a different channel. By translating this condition to bandwidth requirements, we obtain:

$$b_{d,f} \geq b_s. \tag{14.4}$$

To implement this condition, when a peer begins uploading a new primary stream, it automatically increments the number of reserved streams, $n_{d,r}$ until the stream is no longer uploaded or changed to secondary at the next channel change. The rationale behind the upload of primary streams is the increased flexibility in the assignment of network resources, and the self-organization of peers. Whereas in VUD, the peer participation on the uploading (secondary) streams is determined beforehand, in a centralized fashion, as shown in the figure 14.2(a), the implementation of the PCA accommodates the uploading on an on-demand basis, as illustrated in the figure 14.2(b). The organization of the streaming overlay is *decentralized*, meaning that peer, through the selection of their viewing TV channel and the channel demand, influence the setup of secondary streams.

The figure 14.3 illustrates, in further detail, the mechanism involved in the upload of a primary stream. In the initial state, at step 1, a given UE peer receives four primary streams, for the current TV channel, and three secondary streams. At step 2, upon receiving a request to upload a primary stream, the peer increments the number of reserved streams. This reservation accommodates the extra stream needed in a situation where the upload would continue past the following channel change. The peer flags the primary session, in order to process it differently when the user changes the current TV channel.

At the optional step 3, depending on the configuration parameter called `PolicyEarlyCommit`, the peer may commit the reserved session immediately, by establishing an inactive download session, similar to an inactive upload one. Alternatively, the UE can delay the establishment until the new session becomes necessary. The first option has the advantage of a shorter channel change delay, while the second conserves the network bandwidth[2].

Finally, at the step 4, the user changes the TV channel. In the example, we assume that the upload of the primary session continues at least until this moment. In this situation, the flagged primary session becomes secondary, continuing the upload. The peer uses the new primary stream

[2] If using H.264/SVC or another scalable codec with the policy `PolicyEarlyCommit` disabled, when a channel changes requires one or more new primary sessions, the P2PTV-AS will accommodate the basic layer(s) on the existing sessions, while assigning the enhancement layer(s) to the new sessions.

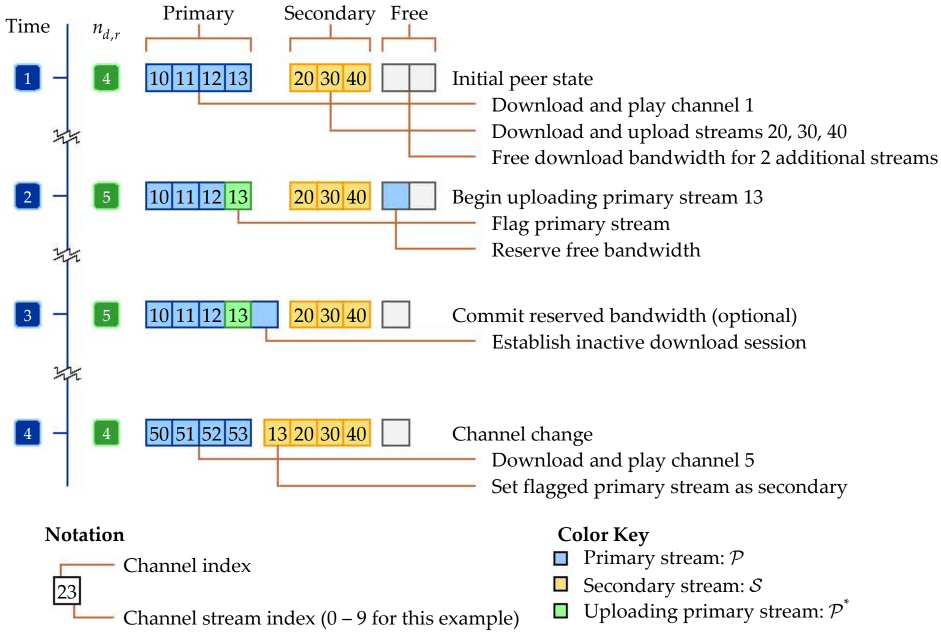

Time
$n_{d,r}$
Primary
Secondary
Free
1
4
10 11 12 13
20 30 40
Initial peer state
Download and play channel 1
Download and upload streams 20, 30, 40
Free download bandwidth for 2 additional streams
2
5
10 11 12 13
20 30 40
Begin uploading primary stream 13
Flag primary stream
Reserve free bandwidth
3
5
10 11 12 13
20 30 40
Commit reserved bandwidth (optional)
Establish inactive download session
4
4
50 51 52 53
13 20 30 40
Channel change
Download and play channel 5
Set flagged primary stream as secondary
Notation
Channel index
23
Channel stream index (0 – 9 for this example)
Color Key
Primary stream: $\mathcal{P}$
Secondary stream: $\mathcal{S}$
Uploading primary stream: $\mathcal{P}^*$

Figure 14.3: The steps involved during the upload of a primary stream.

to support the download of the next channel, assuming the channel requires the same number if streams. If the `PolicyEarlyCommit` parameter is not set, the peers establishes a new downloading SIP session at this time.

Code sample 14.1: The decentralized criteria used for stream uploading.

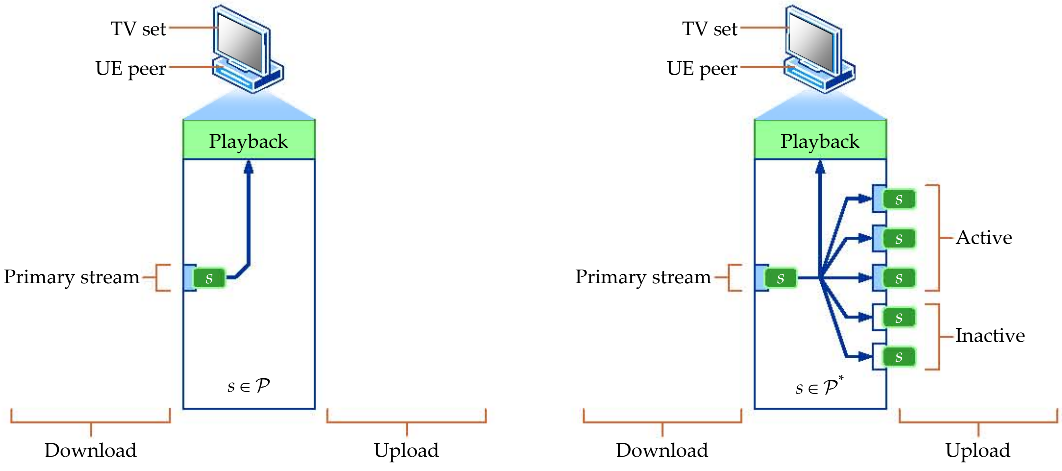
```
% Input:
%   s   - the stream
% Output: whether the peer may upload the stream s
function DecentralizedUploadStream(s)
    b_uf = b_u - (n_a + n_i) * b_s; % Compute free upload bandwidth
    b_df = b_d - (n_r + n_s) * b_s; % Compute free download bandwidth
    if b_uf >= b_s % If sufficient upload bandwidth
        if ismember(s, P) && b_df >= b_s
            n_r = n_r + 1; % Increment the number of reserved streams
            P = setdiff(P, s); % Remove stream from set P
            Pstar = union(Pstar, s); % Add stream to set Pstar
            return true; % Can upload primary stream
        elseif ismember(s, S) || ismember(s, Pstar)
            return true; % Always can upload a secondary stream
        else
            return false; % Cannot upload
        end
    else
        return false; % Cannot upload
    end
end
```

On the uploading side, the peer and the PCA uses a similar bandwidth constraint to determine whether there exists sufficient upload bandwidth:

$$b_{u,f} \geq b_s. \tag{14.5}$$

The code sample 14.1 illustrates the simplified Matlab code used to determine whether a peer may upload stream s. We introduced a new set, denoted by $\mathcal{P}^*$ or `Pstar`, to track the uploading primary streams. The two sets are disjoint, such that we always have $\mathcal{P} \cap \mathcal{P}^* = \emptyset$.

The implementation of the production software is substantially more complex than shown



(a) $s \in \mathcal{P}$ (b) $s \in \mathcal{P}^*$

Figure 14.4: The determination of primary stream s set membership.

here. In principle, for each uploading stream, either primary or secondary, the algorithm maps the corresponding uploading streams, and the primary stream is automatically considered a member of the sets $\mathcal{P}$ or $\mathcal{P}^*$, depending on whether the number of mapped streams is zero. Toward this end, the figure 14.2.2 shows the determination of set membership based on the corresponding upload streams.

14.2.3 Centralized Coordination

In addition to the decentralized organization of the peers, which relies on the TV channels selected by users, the P2PTV-AS may also initiate the download of secondary streams. This mechanism is useful when a large fraction of peers watching a specific channel, do not have sufficient bandwidth to support the upload, and additional capability is required.

In VUD, the peer assignment algorithm uses a metric called *resource index*, denoted by ρ_{VUD}, which measures the bandwidth sufficiency of a given stream s as the ratio between the upload capacity available to the stream and download bit-rate[3]:

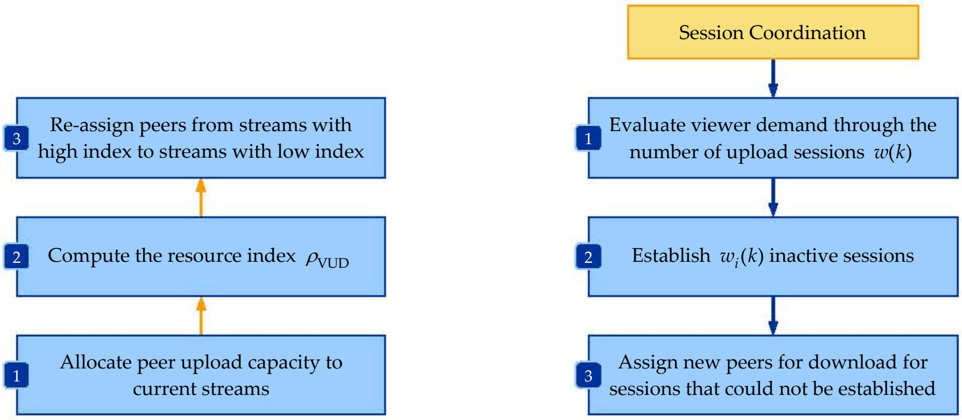
$$\rho_{\text{VUD}}(s) = \frac{\text{upload capacity available to stream } s}{\text{number of viewers} \times b_s}. \tag{14.6}$$

To determine the available upload capacity, the peer upload bandwidth is discreetly divided between the current set of uploaded streams. When an imbalance is detected, the peers and their bandwidth are reassigned from resource-rich to resource-poor streams.

For the design of the PCA, we feel that the resource index approach is unjustifiable complex for the following reasons. The VUD method uses a bottom-up strategy, where peer capacity is divided across current streams, and the resulting resource index metric indicates the optimal re-assignment of peers, such that the available resource satisfy the viewer demand. The philosophy of our decentralized coordination is opposite, a top-down approach, where peer capacity is initially allocated according to demand, and streams are re-assigned when more resources are required. Because the centralized coordination complements the decentralized mechanism for resource-poor streams, we follow a set of similar rules here.

Like before, the SCA plays a key role in estimating the viewing demand. Whereas the VUD algorithm uses the resource index to detect when there is a deficient in upload capacity, the PCA

[3] The original notation used by We et al. uses only ρ, however, we extended their notation to prevent any confusion to the upload session utilization.


3
Re-assign peers from streams with high index to streams with low index
2
Compute the resource index ρ_{VUD}
1
Allocate peer upload capacity to current streams
Session Coordination
1
Evaluate viewer demand through the number of upload sessions $w(k)$
2
Establish $w_i(k)$ inactive sessions
3
Assign new peers for download for sessions that could not be established

(a) View-upload decoupling.

(b) Peer coordination algorithm.

Figure 14.5: Comparison between the peer assignment.

uses the establishment of the inactive upload sessions as an equivalent metric. If the P2PTV-AS cannot establish a new session because the existing peers receiving the stream do not have enough bandwidth, the PCA selects a new peer to download the stream. To this end, the figure 14.2.3 compares our approach against the VUD, emphasizing the differences.

Code sample 14.2: The centralized criteria used for stream uploading.

```
% Input:
%   s   - the stream
% Output: whether the peer may download and upload the stream s
function CentralizedUploadStream(s)
    b_uf = b_u - (n_a + n_i) * b_s; % Compute free upload bandwidth
    b_df = b_d - (n_r + n_s) * b_s; % Compute free download bandwidth
    if b_uf >= b_s && b_df >= b_s % If sufficient upload and download bandwidth
        n_s = n_s + 1; % Increment the number of secondary streams
        S = union(S, s); % Add stream to set S
        return true; % Can download and upload
    else
        return false; % Cannot download and upload
    end
end
```

In the code sample 14.1, we present the simplified Matlab code that determines whether a peer may download and upload stream s assuming the centralized coordination. Unlike the decentralized mechanism, this situation only requires sufficient bandwidth for both download and upload. The new stream is included by default in the secondary streams set, $\mathcal{S}$.

14.3 Algorithm Design

Due to the complexity involved in handling the state of such a large-scale distributed system, the PCA implementation uses a cooperative system, divided between the P2PTV-AS and the UE peers. Namely, the UE peers monitor the available and usage of the network resources and report the resource changes to the application server using a SIP notification. The advantages of this approach are the lower overhead (event-driven as opposed to pooling-based measurement), and the enhanced awareness of unexpected network changes such as faults, outages, etcetera. In the figure 14.6, we present the architecture of the peer coordination algorithm, where, in principle, the operation executes as follows.

- The UE peers use SIP notifications, sent on the P2PTV-AS subscription dialog, to inform

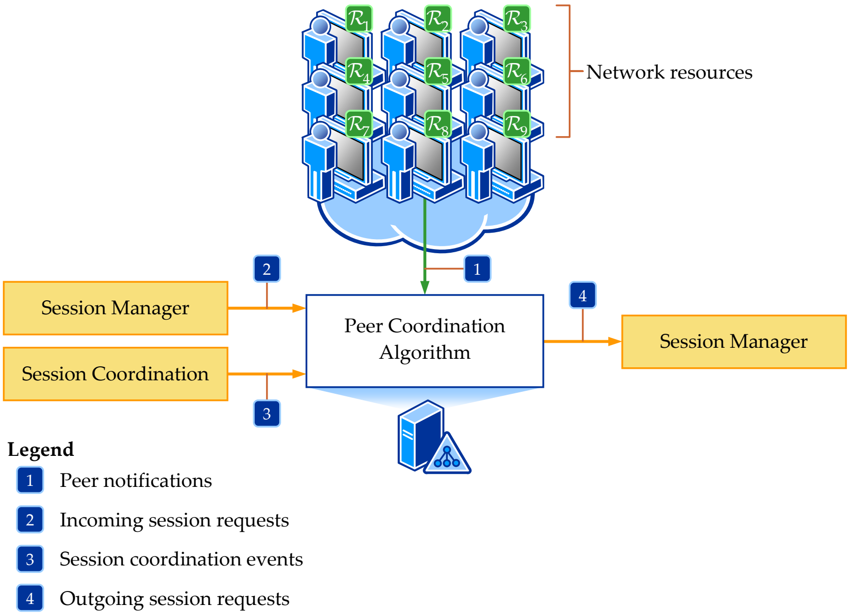


Figure 14.6: The interaction with the peer coordination algorithm.

the PCA of changes in their status (online, offline), network resources, or to require specific actions.

- The PCA receives all initial session requests, for the establishment, update or release of a streaming session. The session manager receiving the actual SIP messages, and maintaining the state of every SIP dialog within the corresponding user agent, communicates these requests via an internal API specific to the implementation.

- The PCA also processes the SCA events, raised at the end of every sampling period and including the estimated number of upload sessions, $w(k)$, for every TV channel.

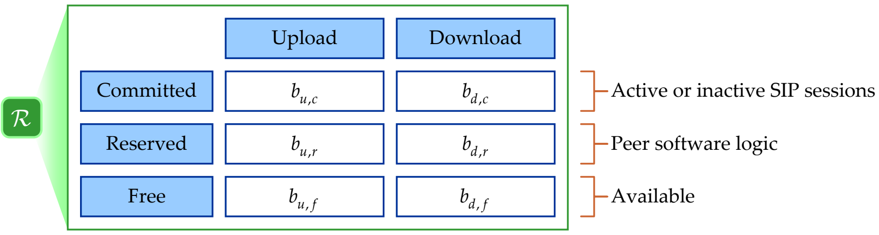
- Finally, the algorithm generates new outgoing session requests, representing the appropriate action to each specific input. Like incoming requests, they are communicated to the session manager using the internal API, which in turn translates them into SIP requests. We note that, with respect to the session manager, the PCA operates at a higher abstraction level, and does not participate in all stages of a SIP dialog.

14.3.1 Peer Data Organization

The PCA uses a light-weight database to maintain a live snapshot of the UE peers and their bandwidth resources, denoted by the set $\mathcal{R}$. At a minimum, this includes the committed, reserved, and free upload and download bandwidth components, as shown in the figure 14.7[4]. Because the P2PTV-AS is the default handler of all multimedia sessions established by the user equipments, it tracks the existing download and upload sessions.

The purpose of the bandwidth resource set is twofold: it enables the PCA to determine, in absolute terms, the UE peers available for new sessions, and it allows the selection of peers according to their capabilities. Given that $\mathcal{R}$ is a multi-dimensional metric, it is not suitable to use it directly for ordering peer resources. For convenience, we introduce the *resource level*, denoted by r, which is a real number and measures the quality of a peer as an uploader.

[4] As future work, it may be possible to include other resource metrics such as latency. In the present work, we focus exclusively on the network bandwidth.



	Upload	Download
Committed	$b_{u,c}$	$b_{d,c}$
Reserved	$b_{u,r}$	$b_{d,r}$
Free	$b_{u,f}$	$b_{d,f}$

Figure 14.7: The network resources included in the peer notifications.

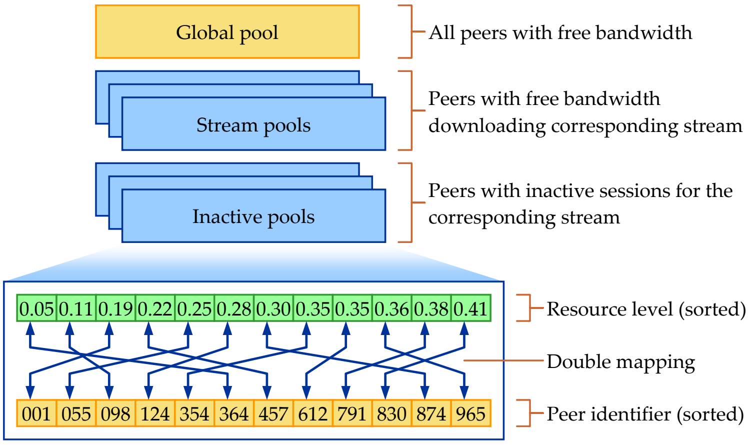


Figure 14.8: The PCA peer pools and pool data structure.

In the interest of minimizing the latency of searching for a new peer, the PCA organizes the peers in three types of peer pools, as shown in the figure 14.8:

- A *global pool*, which includes all peers with *available resources*.
- A set of *stream pools*, one for each stream, similar to the global pool.
- A set of *inactive pools*, one for each stream, similar to the global and stream pools.

In this context, available resources means the peers that meet the decentralized or centralized uploading criteria, as outlines by the algorithms 14.1 and 14.2, respectively.

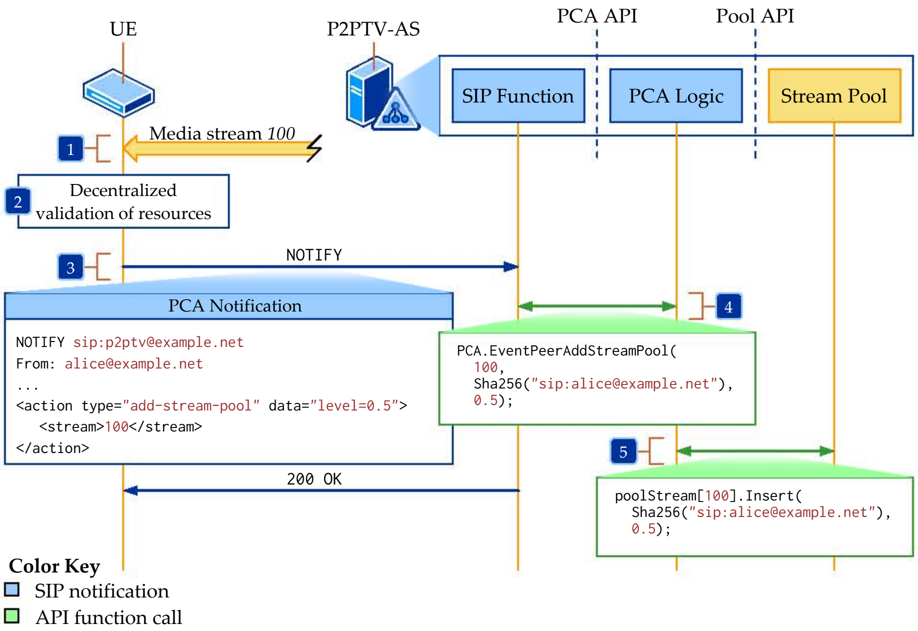
A peer pool is a mapping between peers, represented by numeric identifiers, and their corresponding resource level, r. The peer identifier is specific to implementation, such as a hash of the UE IP address or the public user identity. Both the identifier and resource level tables are kept sorted to optimize the pool operations. In this manner, the insertion, update, and removal of entries is achieved in a logarithmic amortized time, $O(\log u)$, whereas finding the peers with the best or worst resource level is completed in constant time, $O(1)$.

For the global and the stream pools, we propose a resource level that is the ratio between the free and the total average peer upload bandwidth, where a peer with a higher resource level is more desirable:

$$r|_{\text{global,stream}} = \frac{b_{u,f}}{b_u}. \tag{14.7}$$

By using such a metric, our algorithm promotes an equal contribution among peers, relative to their upload capacity. In the inactive pool, the resource level is the number of established inactive sessions:

$$r|_{\text{inactive}} = w_i. \tag{14.8}$$



```
NOTIFY sip:p2ptv@example.net
From: alice@example.net
...
<action type="add-stream-pool" data="level=0.5">
   <stream>100</stream>
</action>
```

```
PCA.EventPeerAddStreamPool(
  100,
  Sha256("sip:alice@example.net"),
  0.5);
```

```
poolStream[100].Insert(
  Sha256("sip:alice@example.net"),
  0.5);
```

Figure 14.9: The interaction between the UE peer and the PCA peer pools.

The rationale behind the organization of peer data in this manner, is to facilitate the cooperation between the peers and the P2PTV-AS. Unlike VUD, where the centralized algorithm assigns peers to streams, in our decentralized approach, the peers are self-organizing and notify their affinity as uploaders for certain streams, in turn updating their membership to the peer pools. The updates occur whenever their current streams and/or bandwidth resources change.

The *global pool* stores all peers, and their resource level, that have sufficient bandwidth to download and upload at least one stream, regardless of their current download/upload status. By using the resource level, the PCA can quickly identify good uploaders in situations such as establishing a new inactive session. The *stream pools*, one for each stream, store peers, and their resource level, that currently download the corresponding stream and have upload bandwidth to upload at least one stream. Finally, the *inactive pools*, one for each stream, store peers (and their resource level) that currently download the corresponding stream and have at least one inactive upload session established for the same stream.

14.3.2 Algorithm Procedures

The peer coordination algorithm executes synchronously with the following events:

- *control updates*, generated by the session coordination algorithm (SCA), indicating $w(k)$ for each TV channel;
- *session requests*, initiated by the UE peers, corresponding to streaming setup, change or release, and;
- *notification requests*, initiated by the UE peers, including the advertisement of peer resources.

The role of the notification requests is of particular importance. To maintain consistency between the UE state and the peer pools, the UE peers manage their pool advertisement using the notification dialog, between UE and the P2PTV-AS. This has the additional advantage of reliving the P2PTV-AS of the computing effort necessary to validate the peer resources and stream membership, and allows peers to take a greater control over their overlay participation.

API function	Event	XML action	Description
EventControl	control	–	Session coordination events
EventPeerStartDownload	session	–	Start downloading a stream
EventPeerStopDownload	session	–	Stop downloading a stream
EventPeerStopUpload	session	–	Stop uploading a stream
EventPeerAddGlobalPool	notification	add-global-pool	Add to the global pool
EventPeerUpdateGlobalPool	notification	update-global-pool	Update in the global pool
EventPeerRemoveGlobalPool	notification	remove-global-pool	Remove from the global pool
EventPeerAddStreamPool	notification	add-stream-pool	Add to the stream pool
EventPeerUpdateStreamPool	notification	update-stream-pool	Update in the stream pool
EventPeerRemoveStreamPool	notification	remove-stream-pool	Remove from the stream pool
EventPeerUpdateInactivePool	notification	update-inactive-pool	Update in the inactive pool

Table 14.2: The API of the peer coordination algorithm.

As an example, the figure 14.9 shows the interaction between the UE and the PCA, when the UE advertises itself in the stream pool. At step 1, the UE peer begins downloading stream 100, for instance as a results of a channel change, which represents the triggering event. At step 2, the peer validates whether it satisfies the bandwidth requirements to upload the stream based on the decentralized criteria, DecentralizedUploadStream, which contains specific tests depending on whether the stream belongs to the sets $\mathcal{P}$, $\mathcal{P}^*$ or $\mathcal{S}$.

During the step 3, the UE peers uses the subscription dialog to notify the P2PTV-AS to the change of its capabilities, with an action of type add-stream-pool and the data containing the peer resource level (e.g. 0.5). At the P2PTV-AS, the SIP function translates the notification request to a function call of the PCA API, corresponding to the XML action type and data (the step 4). Finally, the PCA logic inserts the new peer record in the stream pool (the step 5).

In principle, the previous mechanism applies to all interactions between the UE peers and the PCA. The triggers, at the UE, include any change in the resource set, $\mathcal{R}$, or the downloaded or uploaded streams. In the table 14.2, we summarize all functions from the PCA application programming interface, and their corresponding events.

14.3.3 Peer Selection

In addition to resource tracking and bandwidth assignment, the peer selection is the last main objective of the peer coordination algorithm, and it represents the designation of a UE peer to service an outstanding uploading request. The selected peer must have sufficient bandwidth to support the upload session, should offer a low latency for establishing the new streaming session (*fast signaling*), and a reasonable expectation of reliability (*low churn*).

Ideally, to meet these requirements, the selected peers contribute only secondary streams (low churn), using the inactive upload sessions (fast signaling). In the present work, we do not focus on peer reliability beyond secondary stream participation, such as evaluating the online lifetime. The topic remains a future research area, and in the chapter 17, we include several hints on possible solutions.

At times, when constrained by the available peer bandwidth or by the limitations of the session coordination algorithm, we may compromise and select peers with less ideal characteristics. The tradeoff is the higher signaling delay or peer churn, resulting in streaming interruptions and lower video quality. In the figure 14.10, we summarize the peer selection criteria, and its impact on

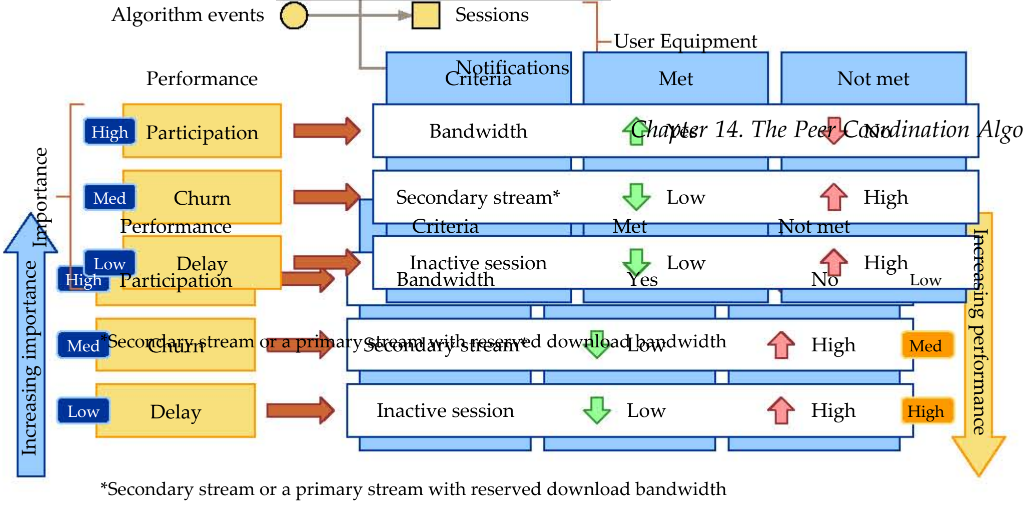


Figure 14.10: The peer selection criteria and its impact on the streaming performance.

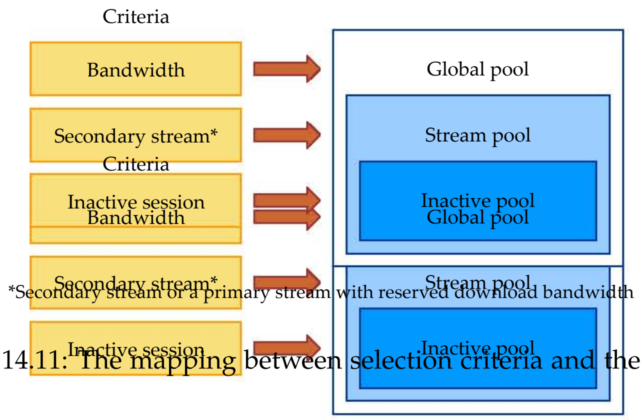


Figure 14.11: The mapping between selection criteria and the peer pools.

the streaming performance. The criteria are ordered by their relative importance. Bandwidth availability, which determines whether a peer may participate in the streaming overlay has the highest importance, whereas the existence of secondary streams and inactive sessions is of medium and low importance, respectively.

We use the peer pools to discriminate between UE peers meeting a subset of the selection criteria. To this end, the figure 14.11 shows the mapping between the selection criteria and the peer pools. Normally, peer pools corresponding to a criterion of a higher performance (e.g. inactive) also include the peers from the lower performance pool (e.g. global). The global pool contains all peers that satisfy the bandwidth conditions; the stream pool adds the peers currently downloading the stream, and finally, the inactive pool includes the peers with inactive upload sessions.

During the selection, the algorithm starts by searching a peer in a high performance pool. If the pool is empty, it continues the search in the lower performance pools. Within each pool, peers are sorted according to their resource level and higher level peers are preferable in the current design. Because the peer pools classify the peers according to their performance characteristic, this approach is extremely efficient and scales well with the subscriber size and the number of channels.

Depending on the situation, not all peer pools are used. For instance, during the selection of a peer to establish a new inactive session it does not make sense to search for a peer in the inactive pool. These peers already have inactive sessions. Likewise, during performance sensitive situations, such as a channel change, the global pool may offer an inadequate performance level, and it is more desirable to use an alternative streaming source.

For the purposes of the peer selection, we identify the following circumstances:

- the connection to a TV channel stream, most often during a channel change;
- the recovery of a TV channel stream, interrupted by a peer departure, and;
- the establishment of an inactive upload sessions.

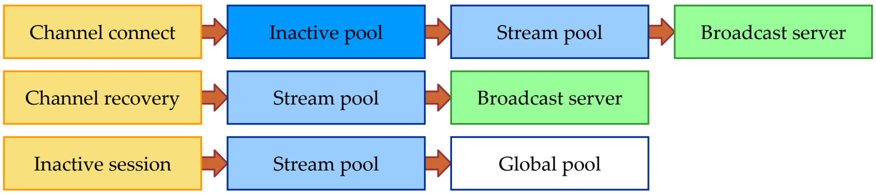


Figure 14.12: The schematic of the peer selection algorithm, depending on the triggering event.

For the reasons stated before, each event requires a different approach. In addition, we include the possibility that the algorithm does not find a suitable uploading peer in any peer pool. In such a circumstance, the P2PTV-AS chooses the broadcast server by default. The figure 14.12 summarizes the access order of the peer pools. When a peer pool is empty, the algorithm selects the next available pool.

14.3.3.1 Channel Connection

The connection to a channel stream, either during the initialization of the user equipment or part of a channel change, requires both fast signaling and churn reduction for optimal playback quality. Therefore, the selection begins from the inactive peer pool, where it chooses the UE peer with the higher resource level, r.

When the inactive pool is empty, such as during the first viewers watching the TV channel or when the session coordination underestimates the demand, the PCA attempts to use peers from the stream pool. Finally, if the stream pool is empty as well, the algorithm chooses the broadcast server, or an equivalent streaming mechanism specified by the service provider. It is notable that connecting to the channel does not require the global pool. The reason is that, while the peer in the global pool have enough bandwidth to participate in the overlay, they do not download the stream in question, and according to the rules of decentralized coordination cannot be uploaders.

14.3.3.2 Channel Recovery

Channel stream recovery is similar, with a major exception: the exclusion of the inactive peer pool. The rationale of this omission lies in the limitations of our current design. Namely, the session coordination algorithm, which determines the number of inactive sessions that accommodate the current channel viewers, does not account for the streaming interruptions caused by peer churn. When including churn, the SCA underestimates the real inactive session demand.

A possible solution is redesigning the session coordination, to include the peer coordination system in the feedback control loop. Toward this end, in the chapter 17, we present a possible approach to the this problem. However, since the peer churn is mitigated to a great extent by the upload through secondary streams, we believe that an additional level of protection does not justify the overall complexity.

As a compromise, the P2PTV service provider may tune a configurable parameter to force the assignment of foster peers with inactive sessions during channel recovery. This configuration option, named `policyRecoverInactive`, enables a greater flexibility, and when used the inactive sessions are shared between channel changes and recovery procedures. In the next chapter, the performance evaluation will analyze in greater detail the impact of this policy.

14.3.3.3 Inactive Sessions

At the beginning of each sampling period, the PCA compares the number of sessions with the output of the session coordination algorithm. When more inactive sessions are needed, the peer selection searches the stream and global pools, respectively. The former relies on the decentralized coordination to benefit from peers already receiving the stream, while the latter uses the centralized coordination to select, similar to VUD, the UE peer with the highest resource level.

14.4 Performance Evaluation

We perform a preliminary evaluation of the peer coordination algorithm to determine its performance characteristics. In particular, we are interested to test the bandwidth assignment based on the decentralized coordination, the main feature of our work that sets it aside from view-upload decoupling (VUD) (D. Wu, Liang, et al., 2009). Before we begin, we feel that it is necessary to mention that these results are limited in scope, and intend only to validate the main features of the PCA. The part V of this thesis analyzes in a greater depth the performance of the P2PTV service.

14.4.1 Experimental Setting

Toward this end, we implemented the peer coordination algorithm in a customized, large scale peer-to-peer simulator. The TV viewer activity is governed by our synthetic IPTV workload, based on the model derived from the Qiu et al. experimental data (Qiu, Ge, Lee, Wang, Xu, & Zhao, 2009). The algorithm receives control events from the session coordination algorithm, representing the necessary number of upload sessions, $w(k)$, for a desired blocking ratio $\beta = 0$, and where the inactive sessions are determined as:

$$w_i(k) = w(k) - u(k). \tag{14.9}$$

In addition to the workload characteristics, the experimental setting has three dimensions:

- peer bandwidth;
- algorithm configuration, and;
- channel encoding.

14.4.1.1 Peer Bandwidth

For an effective assessment of our algorithm's performance, selecting an appropriate but realistic bandwidth distribution across peers is essential. As mentioned previously, in the chapter 4, we envision a deployment where the P2PTV service uses the difference between the link bit-rate and the bandwidth reserved for Internet connectivity. To this end, we test three different scenarios in the increasing order of peer uplink and downlink bandwidth.

- *Poor*, where the UE peers have ADSL2 or ADSL2+ last mile connectivity. Peer upload rate ranges from 544 kbps and 2.58 Mbps as the difference between the line rate and their Internet connectivity package. Download rates range in 10–15 Mbps.
- *Middle-class*, where the UE peers have ADSL2+ connectivity, and the P2PTV service uses the difference to the subscriber's Internet package. Upload rates are 2.58–3.32 Mbps, download rates are 14.5–23.5 Mbps.
- *Rich*, where the UE peers have a rich last mile connection (such as fiber-to-the-building, fiber-to-the-premise, DOCSIS, VDSL, etc), that offer an upload rate significantly higher than ADSL. In this case, we selected an arbitrary upload rate at 10 Mbps, and a download rate at 20 Mbps for all peers, which is sufficient to accommodate all requests.

Fraction	Poor		Middle-class		Rich	
	Download	Upload	Download	Upload	Download	Upload
15 %	15 000	544	23 576	3 328	20 000	10 000
20 %	13 000	800	21 576	2 944	20 000	10 000
50 %	10 000	2 584	18 576	2 548	20 000	10 000
15 %	12 000	2 584	14 576	2 548	20 000	10 000

Table 14.3: The bandwidth scenarios used for the experimental evaluation (in kbps).

The bandwidth is allocated to Internet connectivity as follows: 15 % with 1 Mbps downlink/256 kbps uplink, 20 % with 3 Mbps/640 kbps, 50 % with 6 Mbps/1 Mbps and 15 % with 10 Mbps/1 Mbps, as a typical ADSL access scenario in Europe (Bikfalvi et al., 2011). The table 14.3 summarizes the peer bandwidth scenarios that we use in our experimental evaluation. Given the stream bit-rate requirement of 250 kbps, we expect that the poor and middle-class situations will have a limited upload capacity, and the broadcast server will play a greater role in the content uploading.

14.4.1.2 Channel Specifications

All 700 TV channels contain standard-definition (SDTV) video, with a bit-rate of 2 Mbps, divided in 8 streams of 250 kbps each. For the purposes of our evaluation, we assume that the video bit-rate includes any lower layer overhead required for the transmission. We do not specify an audio-video encoding, because in this chapter we focus on the behavior of the PCA at the stream level, as opposed to the channel-level performance.

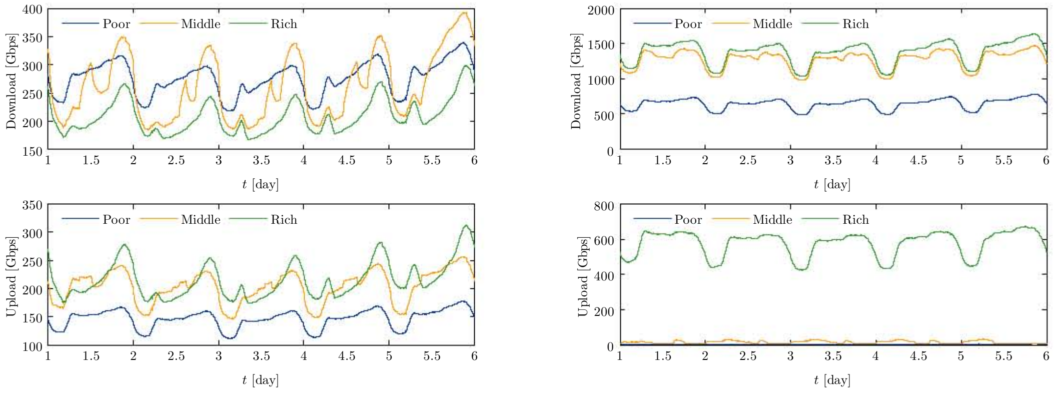
14.4.1.3 Algorithm Configuration

The configuration of the peer coordination algorithm plays a key role in the performance characteristic of the system, in addition to the peer bandwidth. In the table 14.4, we summarize the main policy flags, which can be tuned by the service provider at the P2PTV-AS. These enable a greater control over the allocation of resources, by adjusting the aggressiveness toward bandwidth reservation and usage of the inactive sessions.

Policy	Flag	Default	Fast Download	Fast Recovery	Greedy
Early commit	`PolicyEarlyCommit`	Disabled	Enabled	Disabled	Enabled
Recovery to inactive	`PolicyRecoveryInactive`	Disabled	Disabled	Enabled	Enabled
Finish upload state	`PolicyUploadSession`	Inactive	Inactive	Inactive	Inactive
Download reservation	`PolicyDownloadReserve`	8	8	8	8
Download threshold	`PolicyDownloadThreshold`	250	250	250	250
Upload threshold	`PolicyUploadThreshold`	250	250	250	250

Table 14.4: Configuration flags that adjust the configuration of the peer coordination algorithm.

The `PolicyEarlyCommit` and `PolicyRecoveryInactive` policies have been introduced earlier in this chapter. The first controls whether the PCA reserves bandwidth for an additional download stream, up to a configurable `PolicyDownloadReserve` limit, when uploading a primary stream. This behavior ensures the availability of an inactive download session, if the TV viewer changes the channel before the completion of the upload. The second policy permits the PCA, when enabled, to use the inactive upload sessions of a TV channel for the accommodation of orphaned peers due



(a) Committed peer bandwidth.

(b) Free peer bandwidth.

Figure 14.13: The absolute value UE peer bandwidth, in time domain.

to churn. In certain circumstances, this approach enhances the delay performance of a session recovery, at the cost of fewer inactive sessions available to channel changes.

The PolicyUploadSession flag indicates the state of a peer upload session, when the corresponding receiving party closes the download. When set to *closed*, the P2PTV-AS terminates the session, releasing the committed network resources. If set to *inactive*, the session is maintained and the peer joins the inactive pool for the corresponding stream.

Finally, the last three flags fine-tune the PCA operation: PolicyDownloadReserve specifies the maximum number of reserved download streams, which are allowed when the early commit policy is enabled; PolicyDownloadThreshold and PolicyUploadThreshold represent the minimum free peer download and upload bandwidth, in kbps, used for both the decentralized and centralized upload criteria.

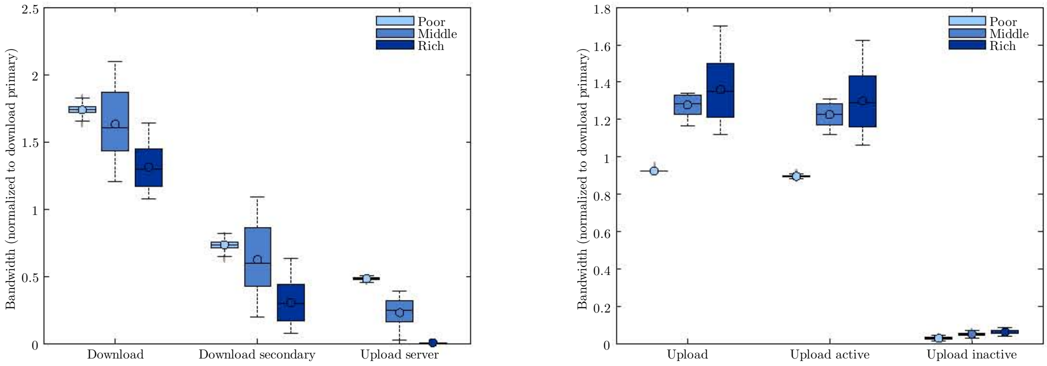
For the preliminary evaluation, the value of the last four flags is constant, as shown in the table, where the selection is consistent to our previous discussion. Depending on the setting of the first two flags, we identify the following configuration scenarios: *default*, *fast download*, *fast recovery* and *greedy*.

14.4.2 Impact of the Peer Bandwidth

The key concept behind the P2PTV service is to leverage the UE unused bandwidth resources to support the uploading of requested TV channels. Toward this end, the bandwidth availability plays an essential role in the performance of the PCA, an by extension in the overall performance. In bandwidth-sufficient situations, the uploading may be supported entirely by the peer population, except for the seed streams supplied by the broadcast server. When resources are scarce, the server must accommodate a larger number of requests.

We simulate the operation of the algorithm, based on the bandwidth scenarios defined previously: *poor*, *middle-class* and *rich*. The total number of P2PTV subscribers is 100 000, watching P2P television during one week. For enhanced relevance of the experimental data, we omit from our analysis the first and last days of the simulation, counting for the warm-up and cool-down periods.

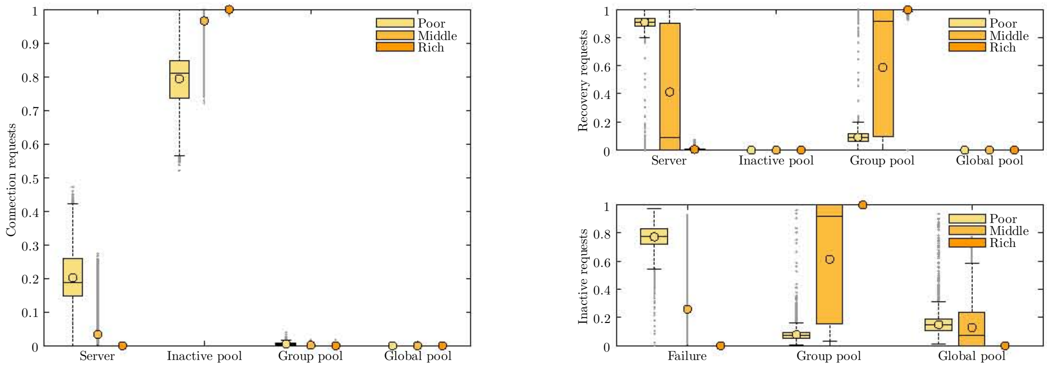
First, in the figures 14.13(a) and 14.13(b), we show the actual measured peer bandwidth, in both uplink and downlink directions. The results are charted against time, due to the diurnal pattern of the user activity, which makes a statistical analysis difficult. We discriminate between the committed and free bandwidth, illustrated in the left and right plots, respectively. We summarize the key findings as follows.



(a) Peer download and server upload.

(b) Peer upload.

Figure 14.14: Bandwidth statistics, normalized to the bandwidth of all primary streams.



(a) Connection requests (normalized).

(b) Recovery and inactive requests (normalized).

Figure 14.15: The servicing of peer coordination algorithm requests.

- All bandwidth scenarios are unconstrained with respect to the downlink, because there always exists free download bandwidth. On the uplink direction, the poor and middle-class cases are constrained, where the free upload bandwidth is near zero, while the rich deployment is not.
- The utilization of the download bandwidth increase for resource poor scenarios, where peers do not have sufficient upload capacity to support many upstream neighbors. Hence, more peers have to download the same content as secondary streams.
- The middle-class peers have nearly sufficient, but slightly less, of the upload bandwidth committed in the rich case. Generally, the peer resources fall short during the peaks of increased user activity, such as evening demand.

To assess the potential of the peer participation, with respect to the real streaming demand, in the figures 14.14(a) and 14.14(b), we illustrate the bandwidth components normalized to the bit-rate of the primary streams. These are the reference showing the actual channel consumption by P2PTV viewers. We remind the reader, that VUD uses a similar performance metric, the resource index ρ_{VUD}, dividing the total upload to content viewed. The main findings are the following.

- The performance of total peer download is supra-unitary, the overhead counting for the secondary streams (the figure 14.14(a)). As noted before, the overhead diminishes for rich peers, where they can accommodate more neighbors.

- The broadcast server contribution is variable across different bandwidth scenarios: for the poor situation it stands at 50 %, 25 % for the middle-class and almost zero when peers are rich. In the last case, the server uploads only the seed streams.

- On the uploading side (the figure 14.14(b)), poor peer upload less (sub-unitary) than what they download as primary streams. Middle-class and rich peer stand at a supra-unitary ratio of approximately 1.3 and 1.4, respectively, where the difference accounts for the upload of secondary streams.

- The majority of the upload sessions are active, with the fraction of reserved but inactive bandwidth standing at less than 10 % in the richest case. This demonstrates the effectiveness of the session coordination in correctly estimating the viewer demand.

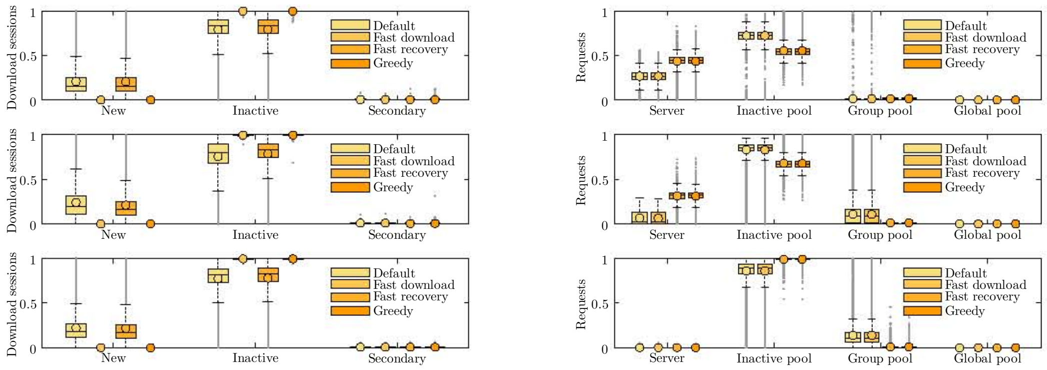
Finally, the figures 14.15(a) and 14.15(b) illustrate the handling of streaming requests by the PCA. The results show, indirectly, the overall setup delay when using P2P media connections, depending on the pool selection. In the table 14.5, we specify the session modifications needed on both the download and upload parts of a target UE peer: *setup* stands for establishing a new session, the *update* uses an existing session, while *none* does not require any changes.

Request	Inactive	Group	Global	Server
	Upload session			
Connection	Update	Setup	Setup	Setup
Recovery	Update	Setup	Setup	Setup
Inactive	–	Setup	Setup	–
	Download session			
Connection	None	None	Setup	None
Recovery	None	None	Setup	None
Inactive	–	None	Setup	–

Table 14.5: Session changes at a target peer, selected from a peer pool.

The majority of the connection requests, when connection to a new TV channel, are accommodated by peers with inactive upload sessions. The fraction of requests begins at 80 % when using poor peers, and reaches almost 100 % in the rich scenario. The remaining requests are largely serviced by the broadcast server, due to the limited peer capacity. Only a very small fraction, around 1 % for poor peers, is handled by group pool peers. The global pool peers are not used for new connections.

Because in the default configuration the inactive pool is exclusive for new connections, churn recovery requests are divided among the server and group pool peers, depending on the bandwidth situation. The requests to establish new inactive sessions are divided across group and global pool peers, taking advantage of the decentralized and centralized coordination, respectively. When resources are scarce, the peer selection algorithm exhibits a high failure ratio, of almost 80 % for poor peers. We note that these failures do not affect the streaming quality, but rather indicate that the peers do not have the capacity to accommodate new requests using inactive sessions. In turn, these requests are serviced using alternative means, such as the broadcast server.



(a) The servicing of connection requests: poor (top), middle-class (middle), rich (bottom).

(b) The servicing of uploading requests: poor (top), middle-class (middle), rich (bottom).

Figure 14.16: The impact of algorithm configuration on requests servicing.

14.4.3 Impact of the Algorithm Configuration

The previous results illustrated the performance characteristic of the PCA, given its default configuration. Alternatively, the P2PTV service provider may use the configuration flags, to fine tune the desired behavior given the setting of a particular deployment. In this section, we examine the impact of this configuration. We distinguish between the following cases: *default*, *fast download* based on the `PolicyEarlyCommit` flag, *fast recovery* based on the `PolicyRecoveryInactive` flag, and *greedy* using both.

The primary benefit of the non-default settings is the fast signaling on the download or upload sessions, during connection and recovery requests, respectively. In the figure 14.16(a), we begin by showing the characteristic of the download session while connecting to a channel stream. Depending on the state of the download dialog before the connection, we have the following possibilities:

- a *new* session requires the establishment of a new SIP dialog, and the reservation of corresponding bandwidth;
- a *inactive* session uses an existing SIP dialog with committed but unused resources, and;
- a *secondary* session means the peer already downloads the target content as a secondary stream.

When the `PolicyEarlyCommit` flag is set, in the fast download or greedy configurations, all connection requests are serviced by existing SIP dialogs. This result opposes the default setting, in which approximately 20 % of downloads require the establishment of a new session.

The figure 14.16(b) illustrates the impact of the recovery to inactive policy. When enabled, both the connection and the recovery requests attempt to use inactive upload sessions from the existing foster peers. This increases the performance only in the rich scenario, where the margin of unused inactive sessions are normally sufficient to accommodate the additional load. In resource constraint situations, the competition for the same resources worsens the behavior, by transferring the requests to the broadcast server.

As a side effect to the changes in request handling, in the figure 14.17, we show the changes to the bandwidth usage. Intuitively, the early commit policy uses more peer download bandwidth, whereas the recovery to inactive impacts the server resources. In the case of the latter, the impact may be positive due to the opportunistic usage of the upload sessions. However, the overall trend may be difficult to estimate in different circumstances.

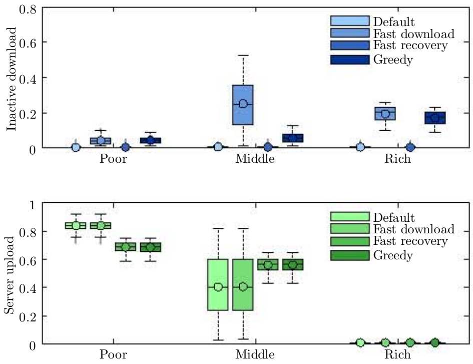
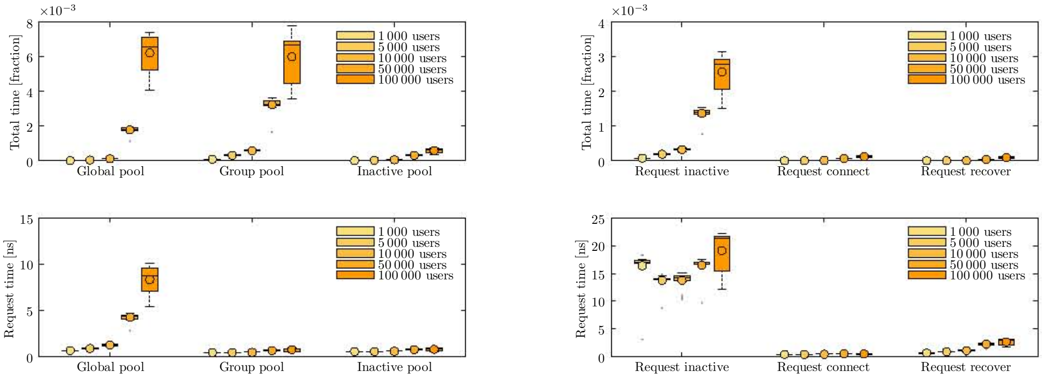


Figure 14.17: The impact of algorithm configuration on bandwidth usage.



(a) Peer pool operations.

(b) Streaming requests.

Figure 14.18: Execution performance of the peer coordination algorithm.

14.4.4 Algorithm Benchmarking

Our peer coordination algorithm provides a simple and efficient mechanism for the management of peer resources and bandwidth assignment. Coupled with the benefits of decentralized uploading, where the UE peers initiate the stream participation based on their current channel selection, significantly reduces the decision steps at the P2PTV-AS.

However, because the PCA is ultimately responsible for the centralized coordination of the available resources, the scalability of the implementation is a genuine concern. Our work included some key concepts, such as the double-mapped peer pools, which are designed with a time complexity of $O(\log u)$ for update and removal requests, and $O(1)$ for peer selection requests.

To measure the behavior of the PCA in real-life, we submit out C++ implementation to an extensive set of benchmarking tests. To increase the accuracy of the measurements, we time the execution of selected API routines handling peer and SCA requests. Unlike the benchmarking of the stochastic control algorithm, where we used the VSTS profiler in instrumentation mode, here we resort to a different approach. A VSTS measurement is not possible because the operation of the PCA relies on I/O exchanges over the SIP signaling with the UE peers to complete a request, which will alter the final result.

Toward this end, we build profiling code directly into the PCA API routines. Although our code is cross-platform, we conduct the measurements on a GNU/Linux platform. Here, the availability of `clock_getres` and `clock_gettime` functions, along with the `CLOCK_PROCESS_CPUTIME_ID` timer and hardware support, makes possible to precisely time the CPU time allocated to a given section of code with nanosecond resolution. The code sample 14.3 illustrates the principle.

Code sample 14.3: Benchmarking code using performance counters.

```
// Measure begin counter
if(clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &timerBegin)) throw ...;
// ...
// Code to benchmark
// ...
// Measure end counter
if(clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &timerEnd)) throw ...;
// Compute the difference
timerDelta = timerEnd - timerBegin;
```

We test five scenarios with 1 000, 5 000, 10 000, 50 000 and 100 000 subscribers, respectively, where all UEs have rich connections such that the peer bandwidth does not have a limiting effect. Each scenario was tested 20 times. The figure 14.18(a) illustrates the execution time of PCA pool operations, that is calls to API functions that insert, update or remove records in the peer pools. In the top plot, we show the total execution time, as fraction of the real-time, using a single thread on a Intel i7 CPU at 3.33 GHz. The bottom result contains the request execution time, in nanoseconds.

Finally, the figure 14.18(b) shows the execution performance of handling peer selection requests. Here, we emphasize the performance of the inactive pool, which absorbs the majority of requests given the bandwidth rich conditions. The peer selection completes in constant time, a fact proven by the request time that does not change with the number of subscribers.

14.5 Conclusions

The peer coordination algorithm, or PCA, is responsible for the management of peer network resources and the allocation of bandwidth upon incoming requests. The requests are generated by the UE peers, when changing the TV channel or when experiencing churn, and by the session coordination algorithm to establish inactive upload sessions.

The key design principle follows the work of a previous proposal called view-upload decoupling (VUD), in which the viewing and uploading streams at each UE peer are different. This allows the user to change their current TV channel, without disconnecting the downstream peers. However, while in VUD the separation between viewing and uploading is complete, in our work we propose a more flexible approach where peers may upload any receiving streams as long as channel-change churn is avoided.

To optimize the bandwidth allocation, the PCA relies on the session coordination algorithm, which is a unique and essential tool to estimate the instantaneous channel demand. Because our system can anticipate demand, when the system is bandwidth-sufficient, the typical response is much faster that VUD, and the peer upload-to-download ratio is close to unitary, a fact confirmed by our experimental validation. Peer participation is triggered in either a distributed or a centralized fashion, taking advantage of the existing downloads.

For an enhanced scalability with large subscriber deployments, the PCA uses a set of peer pools to organize the population of peers according to their current streams and available resources. The approach significantly improves the response delay when coordinating many peers, and the benchmarking results demonstrate that the algorithm execution time is negligible even for large scenarios.

Part V

Performance Evaluation

Chapter 15

Evaluation of the Basic Design

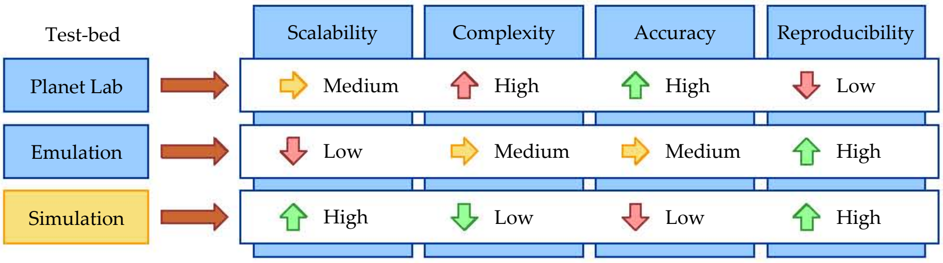
15.1 Overview

Throughout this work, in the interest of a more comprehensive and clearer explanation of our P2PTV service for the IP Multimedia Subsystem, we focused on individual design elements. These included the overall architecture, signaling, mobility issues and the design of the application server software. At each component, we insisted on a performance evaluation as a necessary step to validating the proposed solution.

Although every experimental test indicated that our performance objectives were met, relative to both our goals and to the related work, they did not target the characteristics of our service from the user perspective. Toward this end, in this part and chapter, we conduct an extensive set of evaluation measurements designed to address the existing gap. These results concern the overall behavior of the P2PTV service, versus the existing setting and viewing demand.

We focus on two important characteristics: the *streaming performance* and the *resource performance*. The first intends to assess the playback quality at the user equipment, based on the characteristic of the multimedia session, such as the establishment delay and recovery behavior. The second aspect studies the allocation efficiency of the existing peer resources and the contribution of the broadcast server, relative to the bandwidth scenario.

The experimental setup and the initial conditions play an essential role determining the P2PTV performance. In particular, the centralized nature of the IMS core network and the signaling requirement for the establishment of any multimedia session may introduce a significant performance bottleneck if not designed to accommodate the viewer demand. On the other hand, a large and distributed IMS deployment with many CSCFs, assigned to peers in different networks, will handle P2PTV requests faster, but at the same time may provide a false sense of confidence in the P2PTV service.



Test-bed	Scalability	Complexity	Accuracy	Reproducibility
Planet Lab	⇨ Medium	⇧ High	⇧ High	⇩ Low
Emulation	⇩ Low	⇨ Medium	⇨ Medium	⇧ High
Simulation	⇧ High	⇩ Low	⇩ Low	⇧ High

Figure 15.1: Comparison between possible experimental testbeds.

15.2 Experimental Setting

The objective of this chapter is to conduct a series of experimental measurements, which target a P2PTV deployment using P2P streaming for all TV channels. All TV channels are at standard definition resolution (SDTV), with a constant bit-rate of 2 Mbps divided into 8 equal streams of 250 kbps each. Throughout the experimental evaluation, we maintain a neutral perspective with respect to underlying audio-video codecs, assuming that the channels have the same encoding and bandwidth characteristics, and are accessible to all UE peers. The broadcast server provides the seeding streams and serves as an uploader of last resort, when the peer uploading capacity is exhausted.

Parameter	Notation	Value	Unit
Number of users	U	100 000	–
Number of TV channels	N	700	–
Measurement interval	T	7	days
Measurement warm-up	T_{warm}	1	day
Measurement cool-down	T_{cool}	1	day

Table 15.1: The workload parameters used for the experimental evaluation.

The user activity and the channel popularity are derived from the IPTV workload model that we presented in the part III. The table 15.1 summarizes the setting of the main parameters. The access network bandwidth, available to the user equipments, reuses the *poor*, *middle-class* and *rich* scenarios from the chapter 14. The first two cases correspond to a typical ADSL access network, whereas the last illustrates a situation in which the peer bandwidth is not a limiting factor. The measurement window spans one week of user activity, which encompasses all the major workload characteristics, while providing manageable experimental conditions. In the interest of retaining only representative data, we ignore the first and last days of measurements, corresponding to the warm-up and cool-down periods.

15.2.1 Testbed

Selecting an appropriate testbed to perform the experimental evaluation has a paramount importance. When compared to different areas of networking research, the study of peer-to-peer networks has an intrinsic scalability dimension. This makes the experimental evaluation much more challenging, and requires a careful planning and execution. Toward this end, the figure 15.1 shows the main options available to us:

- real-time experiments in the *Planet-lab*, or an equivalent large-scale, experimental platform;

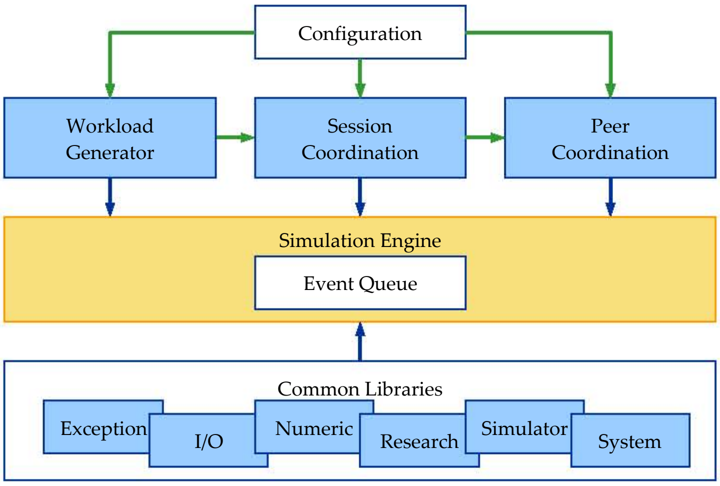


Figure 15.2: The simplified architecture of our event-based simulator.

- real-time experiments in an in-house network *emulation* infrastructure, or;
- virtual-time experiments in a *simulated* environment.

After a careful study of the pros and cons of each method, we argue than a simulation environment provides the best tradeoff. The reasons for this selection include the design stage of the present work, for which the scalability and reproducibility issues are of greater importance. In addition, the main objective of our work – designing a P2PTV service platform for the IMS – is less sensitive to detailed networking aspects, such as congestion and delay. Finally, the assumption that the P2PTV service is deployed in telco walled-garden infrastructure makes the experimental testing over the public Internet problematic.

By considering these tradeoffs, we opted for a simulation platform at this stage of the work. This choice offers a greater and easier control over the experimental conditions. To make the simulation testbed as scalable as possible, we developed our own, in-house, event-based simulation software using the C++ programming language. The accuracy of the simulation is customizable, from a low-level packet-based to high-level session-based. Given reasonable simulation time frames, such as hours, it is possible to simulate a peer-to-peer network with between thousands and hundreds of thousands of peers.

To facilitate the analysis, the experimental data may be directly exported into Matlab for further processing. In the figure 15.2, we sketch the main architecture of our event-based simulator. For enhanced efficiency, the design is largely monolithic, where all simulator modules are compiled as a single executable file. However, the code allows for several layers of abstraction, where real-life entities such as peers, servers, routers or network links are represented by C++ objects, interconnected at runtime through delegated function calls.

A set of libraries provide common functionality. These include C++ exceptions and exception handlers, I/O routines for importing and exporting large data sets; numeric and statistical functions; research models for IPTV workload, protocol prototypes and networking topologies. Simulator functions facilitate the interaction between the higher layers and the simulation engine. Finally, system functions implement optimization features, including multi-threaded execution based on worker threads, and delegate objects to encapsulate function calls.

15.2.2 Simulation Scenario

The topology and the architecture of the IMS infrastructure has a direct influence on the evaluation results. The IMS functional entities, processing the SIP messages, introduce a latency in the

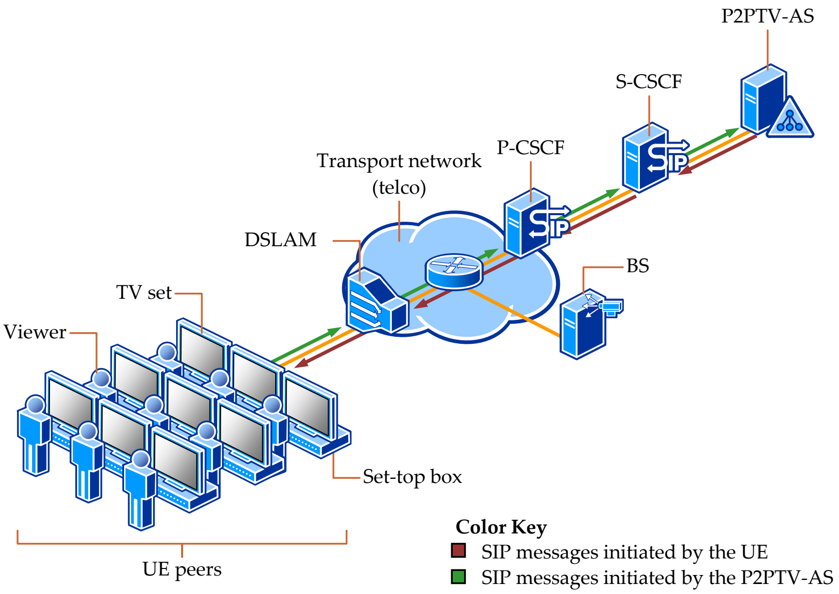

P2PTV-AS
S-CSCF
P-CSCF
Transport network (telco)
DSLAM
BS
TV set
Viewer
Set-top box
UE peers
Color Key
🟥 SIP messages initiated by the UE
🟩 SIP messages initiated by the P2PTV-AS

Figure 15.3: The simulation scenario used in the experimental evaluation.

signaling path. The same is true for non-SIP functions, such as those responsible for policing and charging. Unfortunately, the signaling delay depends on a large number of factors, making it difficult to estimate in practice. For instance, the number of CSCF servers, and the amount of signaling traffic they process, has a significant impact on service behavior.

Due to the difficulties of finding a common ground in a real-life deployment, in our work, we assume the worst-case situation. There exists one P-CSCF and one S-CSCF functional entities, which serve all user equipments, as illustrated in the figure 15.3. At the CSCF level, the signaling messages are queued and processed sequentially, making the overall delay proportional to the instantaneous demand. We do recognize that a practical implementation of the IMS functions may provide a significant number of performance improvements, such as parallel execution, asynchronous I/O operations, worker threads, distributed implementation with load sharing.

The experimental evaluation targets exclusively the delay generated by the establishment of the multimedia SIP session. We consider the non-SIP operations, such as the reservation and commitment of network resources, as out-of-band and executed in parallel with the session signaling. The access network uses ADSL technology, which does not require the initiation of the resource reservation by the UE as in the case of GPRS.

15.2.3 Service Configuration

At last, in the table 15.2, we present the configuration of the P2PTV-AS coordination algorithms. The session coordination uses the coefficients determined in the chapters 11, 12 and 13, independent on the bandwidth scenario. For the peer coordination, we rely on the preliminary evaluation of the algorithm, to customize the parameters according to the bandwidth availability.

Toward this end, we enable the early commit and recovery to inactive policies for all and the rich situation, respectively. These options enhance the signaling performance when the user equipments have ample bandwidth. The download reservation matches the size of a TV channel, while the threshold for peer participation is equal to the bit-rate of one stream.

Configuration	Parameter	Poor	Middle-class	Rich	Unit
Early commit	`PolicyEarlyCommit`	Enabled	Enabled	Enabled	–
Recovery to inactive	`PolicyRecoveryInactive`	Disabled	Disabled	Enabled	–
Finish upload state	`PolicyUploadSession`	Inactive	Inactive	Inactive	–
Download reservation	`PolicyDownloadReserve`	8	8	8	streams
Download threshold	`PolicyDownloadThreshold`	250	250	250	kbps
Upload threshold	`PolicyUploadThreshold`	250	250	250	kbps

Table 15.2: The configuration of the P2PTV-AS.

15.3 Streaming Performance

The evaluation of the streaming performance is a challenging objective. To obtain experimental data that matches the real-life behavior with a high degree of confidence, we must consider the features of our experimental setup. Unfortunately, the ideal metric that quantifies the streaming performance is subjective in nature. After all, the perception of whether the image or sound quality, or the delay experienced during a channel change, are good or bad, depends on the human user.

Given the obvious difficulties in obtaining such a subjective assessment for a large-scale system with tens of thousands of active viewers, we propose a compromising solution. This measures the signaling delay and the peer churn, from where it is possible to infer the effects on the video streaming, and consequently on the viewer opinion. The approach has the additional advantage of being independent from the underlying codec.

15.3.1 Delay Model

Due to the relative novelty of the IP Multimedia Subsystem research, the measurement of the end-to-end delay in such a platform has received far less attention that similar studies in the Internet. Because the IMS call session control functions play a key role in the SIP signaling, one needs to consider carefully the processing delay at these functional entities.

Pesch et al. are among the first authors who address this issue, providing an insight into the IMS platform performance (Pesch et al., 2005). By relying on measurements from a real deployment, their work gives a typical estimate of the CSCF delay, summarized in the table 15.3. We regard this delay as deterministic, and it has the disadvantage that is does not depend on the actual signaling load, as it would be the case in a real situation. Given the measurement difficulties, the authors do not extend their work to studying the delay dynamics or its statistical properties.

Functional entity	Delay	Unit
P-CSCF	25	ms
S-CSCF	25	ms
I-CSCF	25	ms

Table 15.3: The CSCF delay based on the work by Pesch et al.

Recognizing these deficiencies, Ulvan et al. propose an exhaustive model, where the SIP delay between two functional entities is expressed as (Ulvan & Bestak, 2009):

$$D = D_t + D_q + D_p, \tag{15.1}$$

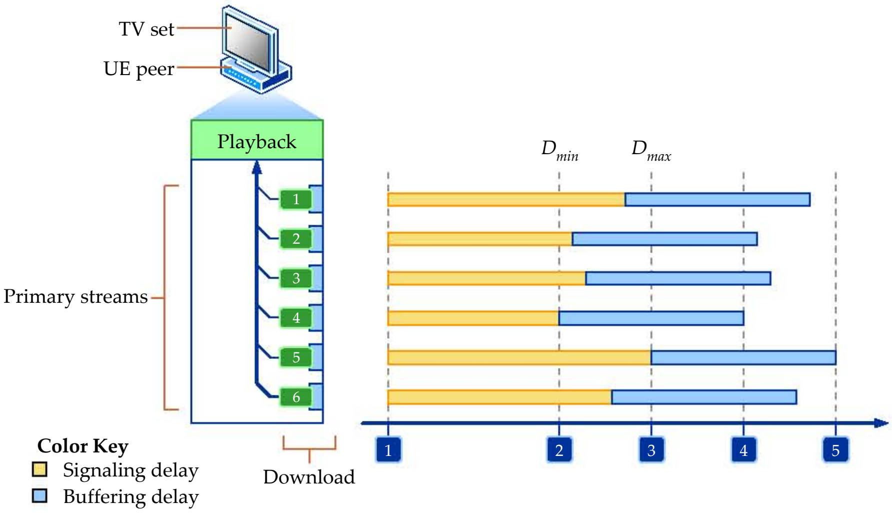


Figure 15.4: The components of the channel connection delay, based on the delay of individual streams.

where:

- D is the total delay;
- D_t is the transmission delay, depending on the message size and connection bit-rate;
- D_q is the queueing delay, depending on the signaling load and processing rate, and;
- D_p is the processing delay, depending on the functional entity and the SIP message.

To simulate the signaling delay in the form suggested by Ulvan et al., we implement the IMS functional entities in the form of a queueing system with a single server. The processing delay is deterministic, and is typically around 4 ms (Munir, 2008).

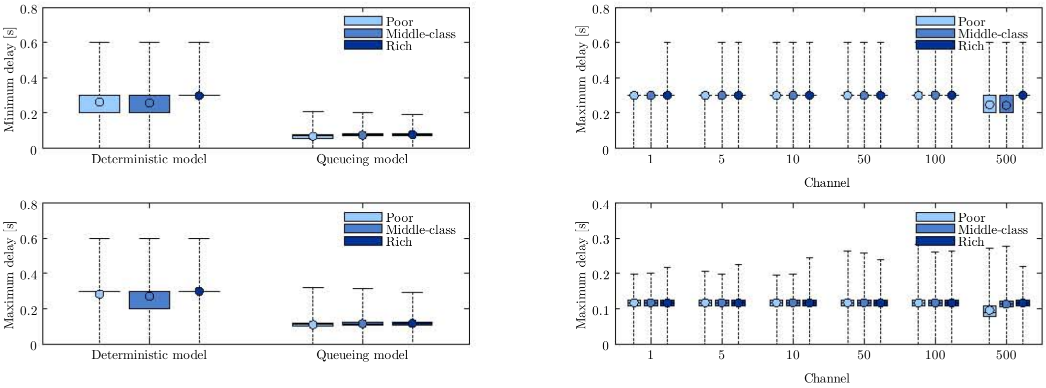
15.3.2 Channel Connection Delay

To be as rigorous as possible, we adopt both delay definitions, *deterministic* and *queueing*. In the first case, the processing of each message at every functional entity takes 25 ms, while in the latter is determines by the queuing sojourn time. We safely exclude the transmission delay, based on the high bit-rates of the core network links.

We measure the signaling delay from the perspective of its impact on the human viewer. To this end, we identify the connection to a TV channel as one of the most important user actions, either when turning on the TV equipment or during a channel change. For instance, turning off the TV set or the delay of the notification messages are much less noticeable, or none at all.

During the connection procedure, the UE establishes multimedia sessions for all primary streams in parallel. The buffering of each stream begins as soon as the session signaling has been completed, using the push-pull mechanism briefly illustrated in the chapter 4. However, the video decoding and playback are conditioned by receiving a number of streams, as specified by the underlying codec. When using SVC, a single stream may be sufficient, and this stream may receive priority handling at the P2PTV-AS such that its signaling completes first. Without scalable video, playback may only begin once all sessions have been established.

Depending on these criteria, we identify a *minimum* and a *maximum* stream delay, denoted by D_{min} and D_{max}, respectively, as shown in the figure 15.4. We note that only measure the signaling delay, starting from the moment of user action (moment 1) and until the completion of the SIP



(a) Channel connection delay: minimum stream delay (top), maximum stream delay (bottom).

(b) Channel connection delay per channel: deterministic delay model (top), queueing delay model (bottom).

Figure 15.5: The channel connection delay.

signaling, at moments 2 and 3. Our results do not include the buffering delay, required for each stream until the beginning of decoding (moments 4 and 5). In practice, this depends on the actual audio-video codec and its minimum data requirements.

The signaling delay depends on the number of IMS functional entities involved during the session setup. The purpose of foster peers and inactive upload or download sessions is to reduce this delay, by minimizing the number of necessary SIP exchanges. The appendix A of this work includes the complete set of signaling procedures, corresponding to each situation. In the table 15.4, we summarize the number of SIP messages received and processed by every functional entity, depending on the initial state of the download and upload sessions (Bikfalvi et al., 2009b)[1].

Download	Upload	d-UE	u-UE	BS	P-CSCF	S-CSCF	P2PTV-AS	Total
New	New	4	4	–	16	16	8	48
New	Inactive	4	1	–	9	9	4	27
New	Server	4	–	2	8	8	5	27
Inactive	New	1	4	–	10	10	5	30
Inactive	Inactive	1	1	–	4	4	2	12
Inactive	Server	1	–	2	2	2	2	9

Table 15.4: The number of SIP messages processed by each IMS functional entity.

Based on the data obtained from computer simulations, the figures 15.5(a) and 15.5(b) illustrate the channel connection delay at the signaling level[2]. The result from the left side shows the setup delay for all TV channels, whereas in the result from the right, we analyze separately the delay for a selected number of TV channels, spread across the popularity spectrum.

We make the following important observations.

- The channel connection delay is highly dependent on the delay model of the IMS functional entities. This outcome is not surprising, and it is one of the main reasons we selected two approaches in implementing signaling processing. Such a behavior may also be encountered in practice, where performance may be differ between various hardware or software.

[1] We denoted by *d-UE* and *u-UE* the downloading and uploading peers, and we note that the SIP-based communication between the P2PTV-AS and the BS, as outlined in the appendix, does not involve any CSCFs.

[2] For the results in this section, we used a modified box plot, in which the whiskers represent the minimum and the maximum of the data set.

- The peer bandwidth influences little the signaling delay, mainly because the fast signaling enhancement is a feature of the session coordination. Because this particular setting is constrained only on the uploading side, if there is insufficient upload bandwidth to setup the inactive sessions requested by the SCA, the peers will use the broadcast server. The delay to the server is slightly lower than to a peer with an inactive session, a fact observed in the smaller average for poor and middle-class peers.

- Both delay models yield an excellent delay value, for both the minimum and maximum stream delay. The actual mean values stand at approximately 300 ms for the deterministic model, and 150 ms for the queueing model, under otherwise ideal conditions. The absolute maximum delays are approximately 600 ms and 400 ms, respectively.

- All TV channels receive a similar allocation of resources, such that the setup delay changes little between channels with different popularity. However, there exists a slight imbalance, in resource poor situations, where for unpopular channels the small number of viewers coupled with bandwidth scarcity increases the utilization of the broadcast server.

15.3.3 Peer Churn

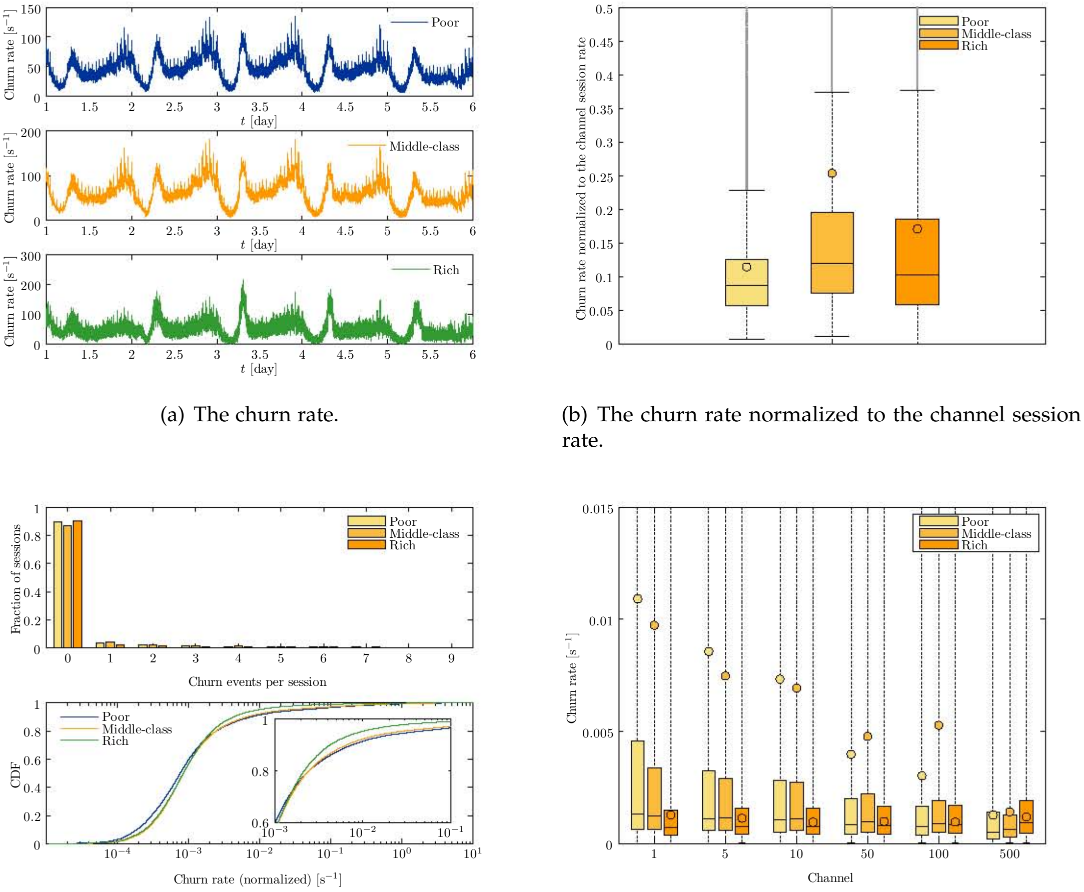
To measure the peer churn, we count the number of sessions recovered while the user watches a TV channel. For graceful peer departures, the P2PTV-AS handles the session recovery before closing the previous upload session. Hence, in normal circumstances, the downloading peer should not experience an interruption in the playback flow. The push-pull mechanism, scalable video codecs and the playback buffer serve as additional mitigation factors.

However, it is important to identify churn events. If the peer churn rate is too high, the streaming flows become highly fragmented, and it will become impossible to maintain a continuous playback without a significant commitment of network and processing resources by all entities involved. Therefore, in the figure 15.6(a), we begin by showing the actual session recovery rate, as measured at the P2PTV-AS. This result, provided as a reference only, illustrates the hourly, diurnal and weekly pattern.

To put the measurement in context, the figure 15.6(b) shows the same recovery rate normalized by the channel session rate. This eliminates the trend of the user activity, and retains only the peer churn relative to the streaming demand. The average peer churn represents between 10 % and 25 % of the session requests, increasing the signaling load by the same factor. The recovery rate is lower for poor and rich peers, where the peer participation is either reduced or more diverse, respectively.

Next, we assess the impact of churn on the user streaming session. We note that each recovery affects only one channel stream, and the effect on the playback depends on the video codec and buffer size. Toward this end, the figure 15.6(c) (top) illustrates the histogram of session churn events. Around 90 % of the channel sessions are churn-free, regardless of the peer bandwidth, although middle-class experience a slightly higher churn for the same reasons as before. For the remaining 10 %, the bottom graph shows the CDF of the churn rate on individual sessions, as experienced by the downloading peers.

At last, the figure 15.6(d) quantifies the churn performance on individual TV channels. Although popular TV channels experience a greater churn rate, we observe that the mean is non-linear with their popularity. In other words, although the viewer demand for the popular channels is several orders of magnitude greater than for unpopular ones, the difference in churn rate is many times smaller. In this case, the higher peer diversity compensates for the greater number of sessions.



(a) The churn rate.

(b) The churn rate normalized to the channel session rate.

(c) Churn performance during a channel session: histogram of session churn events (top) and CDF of the normalized churn rate (bottom).

(d) Churn performance per TV channel.

Figure 15.6: The peer churn performance.

15.4 Resource Performance

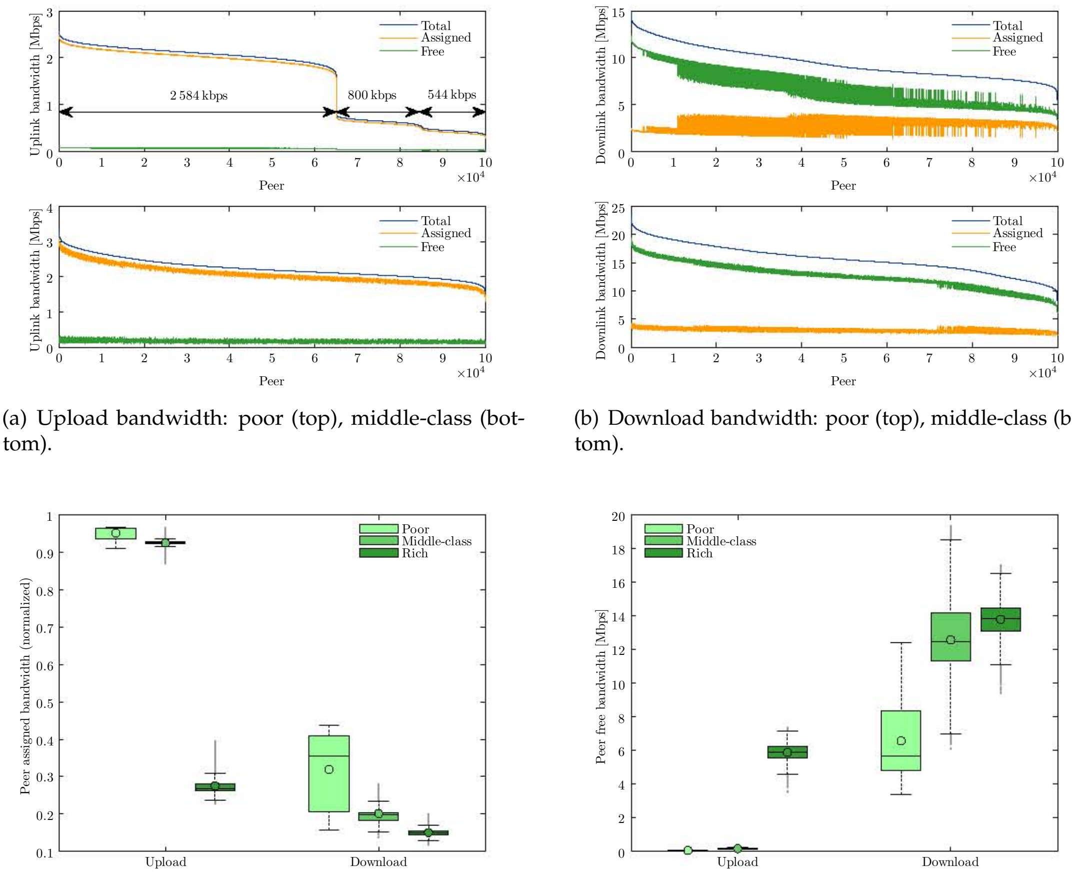
The peer coordination algorithm uses the peer pools to organize the available streaming resources, and allocate them during the peer selection. In order to promote a fair usage among the participating user equipments, the PCA implements the *resource level*, a numeric metric that quantifies the capabilities of each UE, introduced in the chapter 14. This is similar to the resource index from the view-upload decoupling algorithm (D. Wu, Liang, et al., 2009).

The resource usage fairness may be defined in a number of possible ways. In the present work, we proposed a fair conservation of resources, which strives for an equilibrium between the free bandwidth. We allow the rich peers to upload more and poor peers to upload less, as long as we do not deplete the available bandwidth of some peer and leave completely unused the resources of other.

To determine the performance of our P2PTV service with respect to the allocation of resources, in this section we examine the distribution of assigned and free bandwidth, on both uplink and downlink directions. The figure 15.7(a) shows the distribution of the peer upload bandwidth, for poor and middle-class peers[3].

We note that bandwidth is averaged according to the duration the UE is online, and therefore

[3] To keep the results simpler, we do not include the rich scenario in this figure. Rich peers exhibit a similar performance, however because they have identical upload and download capabilities, makes the result less interesting.



(a) Upload bandwidth: poor (top), middle-class (bottom).

(b) Download bandwidth: poor (top), middle-class (bottom).

(c) Committed peer bandwidth, normalized to total peer bandwidth.

(d) Free peer bandwidth.

Figure 15.7: The distribution of peer bandwidth.

UE peers with equal uplink bit rates will show different values, if the total length of their online session is different. In the interest of clarity, the peers are sorted in descending order by their total bandwidth. For poor peers, the distribution clearly marks the boundaries between different bandwidth classes. These are less noticeable for middle-class peers, where the bit-rates are less different.

Despite the differences, the conservation of the free bandwidth is similar across the peer set. The figure 15.7(b), showing the same distribution from the download perspective, yields a similar result. Here, the free bandwidth slightly increases for the poorer peers, because they are usually limited only the uploading side.

Finally, to quantify the fairness of the resource assignment, the figures 15.7(c) and 15.7(d) summarize the previous from results from the perspective of the committed and free peer bandwidth, where the former is expressed as a fraction of the total peer bandwidth. These results emphasize the best the limited upload bandwidth for the poor and middle-class scenarios, as well as the relative uniform distribution of the bandwidth usage across the population of peers.

In the table 15.5, we present the equivalent Jain's fairness index for the previous situations. These values justify the effectiveness of the peer coordination algorithm. Like before, the committed bandwidth is normalized to the total peer bandwidth, while the free bandwidth is compared in absolute terms.

Direction	Poor	Middle-class	Rich
Committed bandwidth			
Upload	0.9996	0.9999	0.9950
Download	0.9109	0.9840	0.9973
Free bandwidth			
Upload	0.9276	0.9808	0.9935
Download	0.9081	0.9674	0.9948

Table 15.5: The Jain's fairness index of the peer resource allocation.

15.5 Conclusions

The performance evaluation from this chapter examines the design quality of the P2PTV service. Whereas in the previous chapters we conducted a set of experiments that address performance issues specific to individual components, here we approach our proposal from the perspective of the viewer and user equipment. Namely, we measure the streaming and resource performance, illustrating the playback quality and the allocation fairness of the peer resources.

The experimental testbed relies on large-scale computer simulations. We believe that this approach offers the best compromise with respect to different options, such as real-life experiments on in-lab network emulation. We reuse the same bandwidth scenarios introduced in the chapter 14, and the P2PTV-AS configuration is customized in each case for best performance.

As opposed to a real-life implementation, the computer simulation requires a model of the IMS infrastructure that captures the delay introduced by the signaling procedures. By relying on the relevant related work, we avoid a significant loss of generality by introducing two delay models: deterministic and queueing-based.

For the streaming performance, we measure the delay and the peer churn. These results capture the effectiveness of our main enhancements, as opposed to using the standardized IMS mechanism of establishing multimedia sessions. The resource performance shows that there exists an almost fair conservation of unused UE resources, despite the complexity of the user activity, peer participation, and heterogeneity of peer bandwidth.

Chapter 16

Extended Scenarios

16.1 Overview

The evaluation of the basic design, conducted in the previous chapter, addressed the performance questions in an experimental setting simple enough to facilitate the identification of the main characteristics. For example, this included the assumption that all TV channels have identical streaming requirements, which is not true in many practical cases.

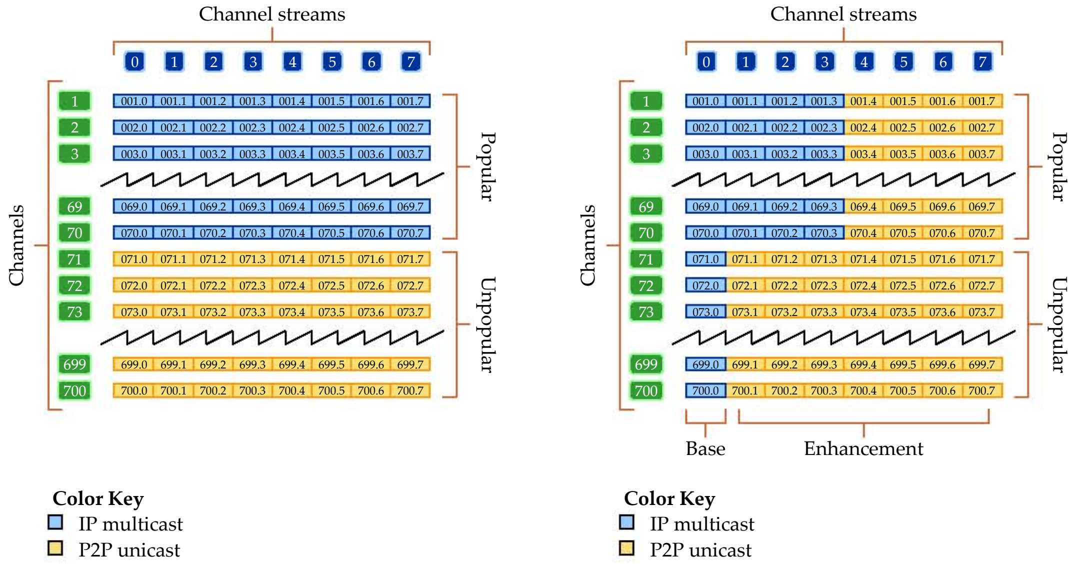
In addition, we imagined a pure peer-to-peer situation, where the broadcast server is reserved as an uploader of the last resort. This mode of operation poses significant challenges in deployments with insufficient peer resources, such as the poor and middle-class scenarios. In such circumstances, a significant fraction of the TV content is uploaded by the server, the same situation peer-to-peer streaming is supposed to prevent.

This chapter extends the performance evaluation by including these two aspects that were previously excluded: hybrid streaming and heterogenous TV channels. The first examines the possibility of using the P2PTV service as a complement to the IP multicast approach. The latter assumes the existence of different types of TV channels, such as standard and high definition, or SDTV and HDTV.

Without a doubt, in reality we may encounter many other different possibilities, which we do not study here in greater detail. In the last chapter of this work, we provide several hints on less important, yet challenging issues, that we missed and provide a great opportunity for further examination.

16.2 Hybrid Streaming

In the previous chapters, the performance evaluation we conducted revealed that access network scenarios, such as ADSL, which are still common in many countries, do not provide us with sufficient network bandwidth to accommodate the viewing demand. In these circumstances, the server utilization remains significant, around 50 % of the downloaded content. Such a behavior,



(a) Channel-level hybrid streaming.

(b) Stream-level hybrid streaming.

Figure 16.1: The concept of hybrid streaming using IP multicast and P2P unicast.

although excellent in terms of the peer bandwidth assignment efficiency, questions the viability of the pure peer-to-peer solution.

When motivating the present work, our main argument against IP multicast, as the classical solution for commercial grade IPTV, included the lack of scalability with an increasing number of TV channels, and the inefficiency of static IP multicast for unpopular TV channels. However, given the limited access network in many real-life situations, we must explore the possibility that the IP multicast solution complements our P2PTV service (Bikfalvi et al., 2011).

This approach results in a *hybrid IPTV* system, where a subset of streams use IP multicast, while the remaining rely on P2P unicast connections. Intuitively, multicast resources are reserved to the streams with higher popularity or of greater importance. This assessment stems from the fact that P2P unicast is always more resource intensive and delay prone, due to the involvement of the user equipments.

We distinguish between the following possibilities: using IP multicast for all or a subset of streams of a TV channel. The figures 16.1(a) and 16.1(b) compare the two possibilities, names *channel-level* and *stream-level* hybrid streaming. The channel-level approach has a coarse granularity, and is well suited for non-scalable video codecs where all streams of a popular TV channel have the same importance. When using scalable video, it is possible to use stream-level hybrid streaming, which assigns IP multicast to the base layer and/or popular channels, while reserving the P2P for the lower-priority enhancement layers.

16.2.1 The Bandwidth Usage

We consider a network topology, depicted in the figure 16.2, comprising of a core network and a number of access networks. The access networks use the ADSL technology, and are connected to the core network through a set of edge routers. The access networks consist of direct links between the edge router and the user equipment (set-top box) from the customer premise. The broadcast server is randomly placed at any core router, and the IPTV[1] subscribers are uniformly distributed across the access networks.

[1] In the present context, we refer to the service as IPTV rather than P2PTV, to emphasize the hybrid streaming.

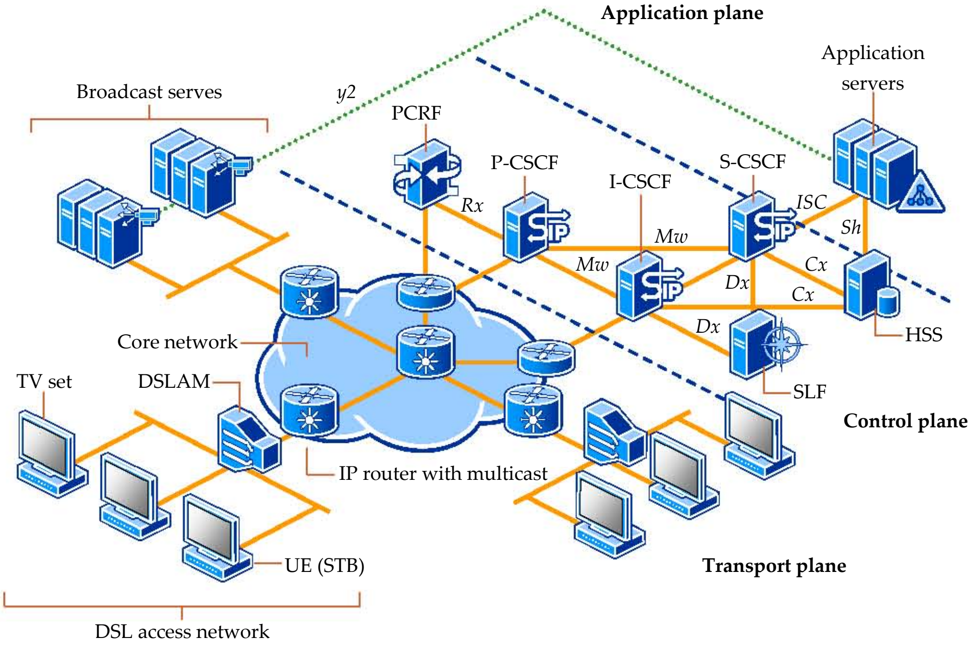


Figure 16.2: An overview of the network topology used for hybrid streaming.

The core network graph is described by a triplet measuring the connectivity and average distance between nodes in P2P unicast and multicast traffic scenarios, shown in the table 16.1. According to previous studies, real networks, such as routing and AS topologies of the Internet, have $m \geq 2$ (Chuang & Sirbu, 2001; Phillips et al., 1999).

Parameter	Description
m	Represents the ratio between the number of routers and links in the network (half of the average node degree).
l_u	Represents the average distance between two random edge routers.
$l_m(v)$	Represents the average size of the shortest-path tree from a random source router to the edge routers of v user equipments.

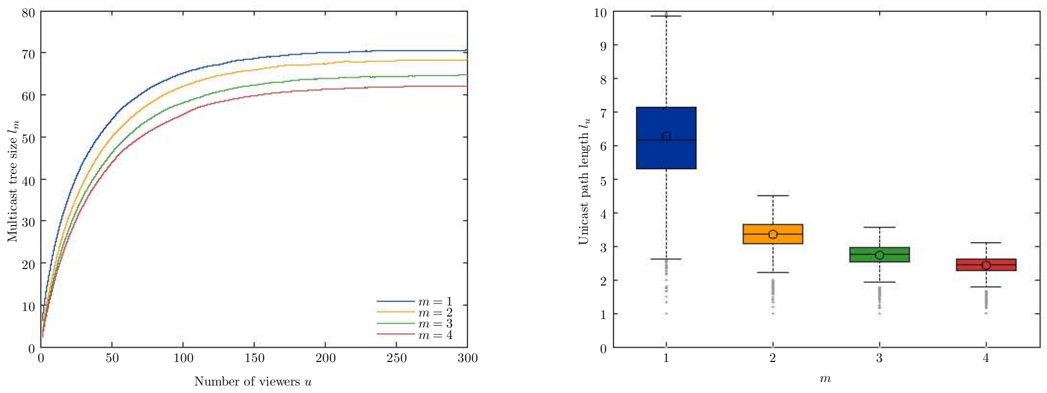
Table 16.1: The triplet describing the core network.

In practice, the multicast tree size, l_m, with respect to the user group size, v, is difficult to determine with a good accuracy. Here, we consider *dynamic multicast*, where the multicast tree is generated in real time, according to the channel requests, using a multicast protocol such as DVMRP or PIM. As indicated by Cha et al., today most walled-garden deployments employ *static multicast* (Cha, Rodriguez, Moon, & Crowcroft, 2008).

For dynamic multicast, the *power scaling law* of Chuang and Sirbu gives a relationship between unicast and multicast (Chuang & Sirbu, 2001). This law introduces k, a multicast scaling factor depending on the network topology, is expressed as:

$$l_m(v) = l_u \cdot v^k, \tag{16.1}$$

and it holds only when the number of users is lower or comparable to the number of edge routers. When the number of users is higher than the number of edge nodes, the tree size enters in a *saturation region*, where it does not increase even if more users are added. The figure 16.3(a) illustrates the relationship between l_m and v in the case of a Waxman-type core network with 100 routers, 50 access routers and $m \in \{1, 2, 3, 4\}$. The equivalent unicast distance, within the core



(a) The relationship between the multicast tree size and the number of viewers.

(b) The unicast distance between the edge router and the broadcast server.

Figure 16.3: Comparison between unicast and multicast distance.

network, is shown in the figure 16.3(b).

When comparing the obtained l_m with their findings, we can observe the average tree size increases with the number of stream viewers, u, up to a limit determined only by the topology and the selection of the edge routers. We denote the saturation limit of the multicast tree size by l_m^∞. For our network topologies, this limit is reached for a multicast group size of more than 250 users. Table 16.2 summarizes the measured values for l_u, l_m^∞ and k for each selected topology. The multicast scaling factor, k, is between 0.8 and 0.9 for the topologies with a higher m, similar to previous findings (Chuang & Sirbu, 2001; Phillips et al., 1999).

m	1	2	3	4
l_u	6.18 ± 0.002	3.30 ± 0.0009	2.70 ± 0.0007	2.40 ± 0.0006
l_m^∞	71.04 ± 0.04	68.65 ± 0.04	64.95 ± 0.04	62.43 ± 0.03
k	0.68	0.82	0.85	0.87

Table 16.2: The set of parameters describing each network topology.

Let us assume that the N TV channels available from the IPTV provider are divided into n individual streams of bit-rate b_s. Among these, a subset of $\mathcal{M}$ streams use IP multicast, and the remaining subset $\mathcal{U}$ rely on P2P unicast connections. We denote by u the average number of active viewers. An individual stream i has the popularity p_i, equal to the popularity of the corresponding TV channel.

The total bandwidth committed in the network, denoted by B, is:

$$B = B_S + B_A + B_C = B_S + B_{A,u} + B_{A,d,m} + B_{A,d,p} + B_{A,d,s} + B_{C,u} + B_{C,m}, \qquad (16.2)$$

where each bandwidth component is illustrated in the figure 16.4. We may approximate the average bandwidth components as shown in the table 16.3 (Bikfalvi et al., 2011). For the analytical bandwidth model, we assume a resource rich situation, where the peer bandwidth does not play a constraining role.

In the previous table, we introduced the performance parameter η that captures the bandwidth overhead introduced by the P2PTV service. This includes both the inactive sessions and the secondary streams, which are used exclusively to enhance the system performance. The value of η is influenced by the user activity and the available peer bandwidth, and is therefore difficult to predict in practice. For our workload and the rich resource scenario, it stands at approximately

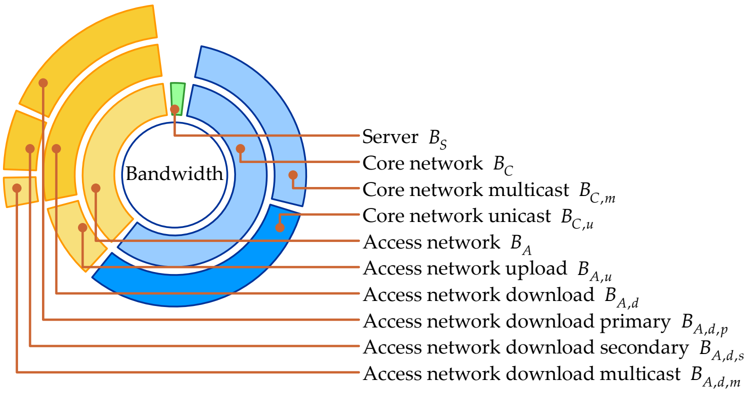

Bandwidth
Server B_S
Core network B_C
Core network multicast $B_{C,m}$
Core network unicast $B_{C,u}$
Access network B_A
Access network upload $B_{A,u}$
Access network download $B_{A,d}$
Access network download primary $B_{A,d,p}$
Access network download secondary $B_{A,d,s}$
Access network download multicast $B_{A,d,m}$

Figure 16.4: The components of the committed network bandwidth.

Component	Equation	Comments
B_S	$\sum_{i\in\mathcal{M}\cup\mathcal{U}} b_s$	In an ideal resource sufficient scenario, the broadcast server uploads each stream once.
$B_{C,m}$	$\sum_{i\in\mathcal{M}} l_m(p_i u) b_s$	The CN multicast bandwidth is proportional to the multicast tree size for each stream, as function of the number of stream viewers.
$B_{C,m}$	$\sum_{i\in\mathcal{U}} l_u p_i u\,(1+\eta)\,b_s$	The CN unicast bandwidth is proportional to the unicast path length for each stream, the number of stream viewers, and the peer-to-peer overhead, η.
$B_{A,u}$	$\sum_{i\in\mathcal{U}} p_i u\,(1+\eta)\,b_s$	The AN upload bandwidth is proportional to the number of stream viewers, and the peer-to-peer overhead, η.
$B_{A,d,m}$	$\sum_{i\in\mathcal{M}} p_i u b_s$	The AN download multicast bandwidth is proportional to the number of stream viewers.
$B_{A,d,p}$	$\sum_{i\in\mathcal{U}} p_i u b_s$	The AN download bandwidth, used for P2PTV primary streams, is proportional to the number of stream viewers.
$B_{A,d,s}$	$\sum_{i\in\mathcal{U}} p_i u \eta b_s$	The AN download bandwidth, used for P2PTV secondary streams, is proportional to the number of stream viewers, and the peer-to-peer overhead, η.

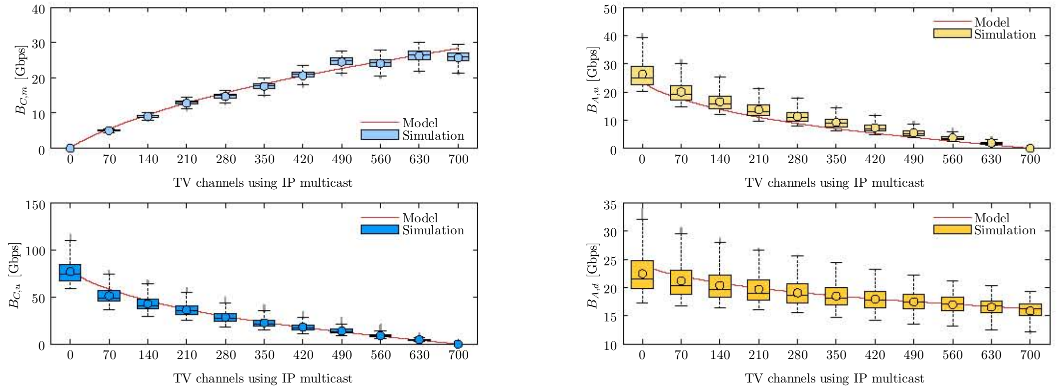
Table 16.3: The analytical model of the committed network bandwidth.

50 %.

To examine whether the analytical model describes well the performance characteristic of the hybrid streaming solution, the figures 16.5(a) and 16.5(b) compare the model and the experimental data, for the core and access network bandwidth, respectively. The hybrid approach uses channel-level streaming, where a selected number of the most popular channels use IP multicast exclusively. We show the resulting bandwidth usage as function of this number. In the interest of making the experimental setup more manageable, given the large number of data points, we limited the number of subscribers to 10 000.

The result validates the match between the analytical model and the simulation, proving that l_m and l_u approximate with a good accuracy the effect of the network topology. At the same time, the constant η measures well the overhead introduced by our P2PTV approach. This result is not surprising because all channels are managed equally by the P2PTV-AS.

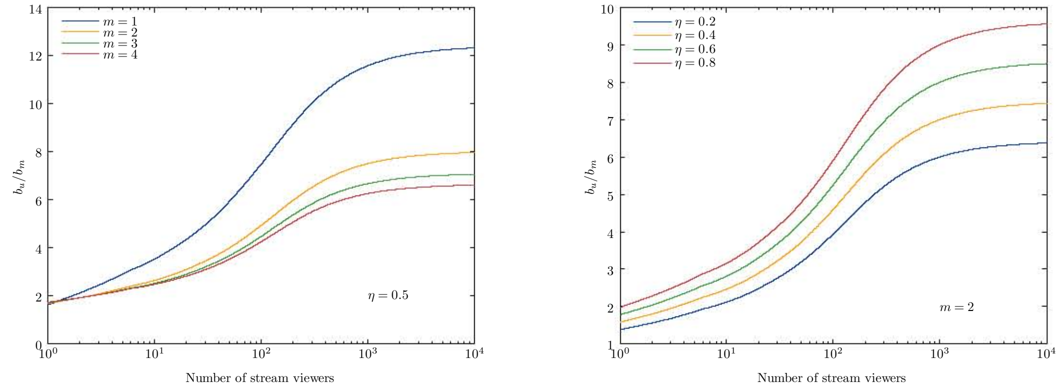
The analytical model helps us understand the behavior of the hybrid streaming in situations that are not covered by our experimental setup, such as a different network topology or activity workload. To this end, the figures 16.6(a) and 16.6(b) examine the ratio between the unicast and multicast bandwidth required by an IPTV stream, denoted b_u/b_m, as function of its popularity.



(a) The core network bandwidth.

(b) The access network bandwidth.

Figure 16.5: Validation of the analytical bandwidth model.



(a) Versus the network topology (m).

(b) Versus the peer-to-peer overhead (η).

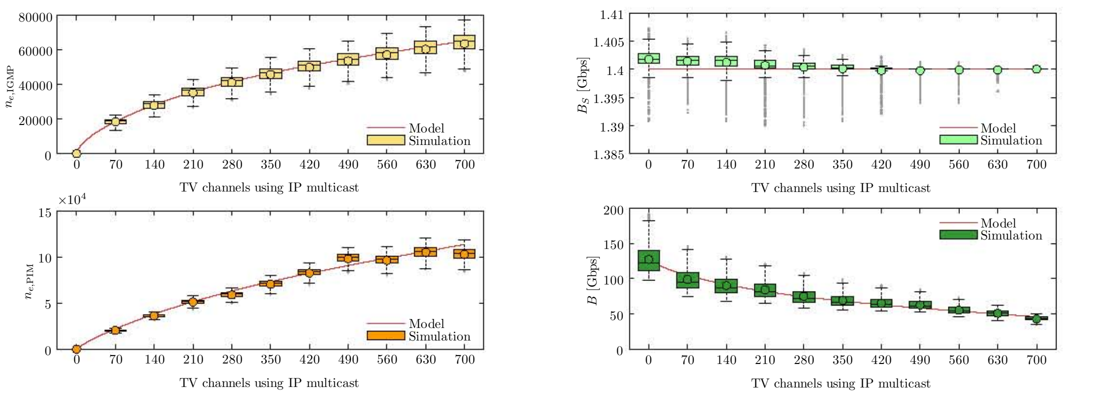
Figure 16.6: Peer-to-peer to IP multicast bandwidth ratio versus the stream popularity.

In the first case, we analyze the impact of different core network topologies determined through the parameter m, whereas in the latter we evaluate the effect of the P2PTV overhead. This result emphasizes that the difference between multicast and peer-to-peer is less significant for unpopular TV channels, for better connected core networks, or when the unicast overhead is smaller.

16.2.2 Scalability Issues

The scalability is one of the multicast issues that has been widely recognized and intensively studied (Wong & Katz, 2000; Thaler & Handley, 2000; Fei et al., 2001; B. Zhang & Mouftah, 2003; Phillips et al., 1999; Radoslavov et al., 1999). When becoming a member of a multicast tree, every router in the network adds a new multicast forwarding entry. However, unlike unicast routing, multicast addresses are not hierarchical and there is no natural way of consolidating multicast entries.

This problem is aggravated for core network routers that will have to handle a very large number of forwarding entries for many TV channels. While the research community has proposed a number of solutions, such as forwarding entries aggregation, where the multicast trees sharing the same interfaces and having a common address prefix are represented by a single entry, still there is no uniformly accepted method. Furthermore, aggregation is not well suited for low popularity TV channels having few users and following many disjoint paths.



(a) Multicast scalability based on the number of routing entries.

(b) Unicast scalability based on the server and total bandwidth.

Figure 16.7: The scalability issues in hybrid streaming.

While we acknowledge that multicast scalability is only a performance problem, which in a hybrid IPTV scenario depends on the number of channels and users, in this section we compare the multicast drawback in terms of scalability. For this purpose, the equations from the table 16.4 estimate the number of multicast entries for IGMP and PIM-SM routers, assuming the point-to-point connection from the user equipment to the edge router. We selected PIM-SM as the multicast protocol most conservative toward the utilization of bandwidth and router entries.

Component	Equation	Comments
$n_{e,\text{IGMP}}$	$\sum_{i \in \mathcal{M}} p_i u$	The number of IGMP entries is proportional to the number of stream viewers.
$n_{e,\text{PIM-SM}}$	$\sum_{i \in \mathcal{M}} l_m\left(p_i u\right)$	The number of PIM-SM entries is proportional to the multicast tree size for each stream.

Table 16.4: The analytical model of the multicast entries.

The figure 16.7(a) examines the analytical estimation of the number of multicast entries against the data obtained from the simulation. As in the case of network bandwidth, we consider channel-level hybrid streaming. The result indicates that the IP multicast poses a greater challenge on the network performance when used for an increasing number of TV channels. Finally, we look at the scalability issues in the peer-to-peer streaming, which are represented by the server and total bandwidth usage, illustrated in the figure 16.7(b).

It is interesting to notice that, when comparing the two scalability dimensions, the changes in terms of bandwidth have a much lesser impact than the number of routing entries, especially for unpopular channels. Therefore, using multicast for unpopular channels brings only a small gain in terms of bandwidth but has almost the same disadvantage for scalability as the popular channels. The server bandwidth usage is particularly unique, and it shows that P2PTV may compete well with IP multicast when the peers have sufficient resources to accommodate the downloading demand.

16.3 Heterogeneous TV Channels

Streaming both SDTV and HDTV channels is possible using our P2PTV service. The problem of heterogeneous TV channels lies in the selection of the participating peers based on whether they meet the channel streaming conditions. When all channels are equal, there exists no distinction.

However, when channels have different bandwidth requirements, the service performance depends on whether on the algorithm is dimensioned for the low or high bit-rate channels.

The key issue of the performance problem is when changing from SDTV to HDTV channels, depending on whether the peers have sufficient free download bandwidth to accommodate the change, or whether that bandwidth is already in use by secondary streams. When peers guard more download bandwidth against the use of HDTV (*conservative*), the changes between SDTV and HDTV are seamless, at the expense of wasting the guarded bandwidth when the cumulative popularity of high-definition channels is low. If peers only reserve bandwidth for SDTV (*flexible*), changing to HDTV is a more expensive process, involving the eviction of some secondary streams, generating churn. A *centrist* approach is also possible, where the UE peers guard a number of downloading streams between the SDTV and HDTV requirements.

Let us assume an experimental scenario, where 5 % of the 700 TV channels are HDTV. Like their SDTV counterparts, their are divided into streams of equal bit-rate at 250 kbps. The table 16.5 summarizes the difference between the channel types. We consider the HDTV channels uniformly spread across the popularity spectrum, such as having high-definition version of SD counterparts.

Type	HDTV	SDTV
Fraction	5 %	95 %
Number	35	665
Bit-rate	10 Mbps	2 Mbps
Streams	40	8

Table 16.5: The specification of the HDTV and SDTV channels.

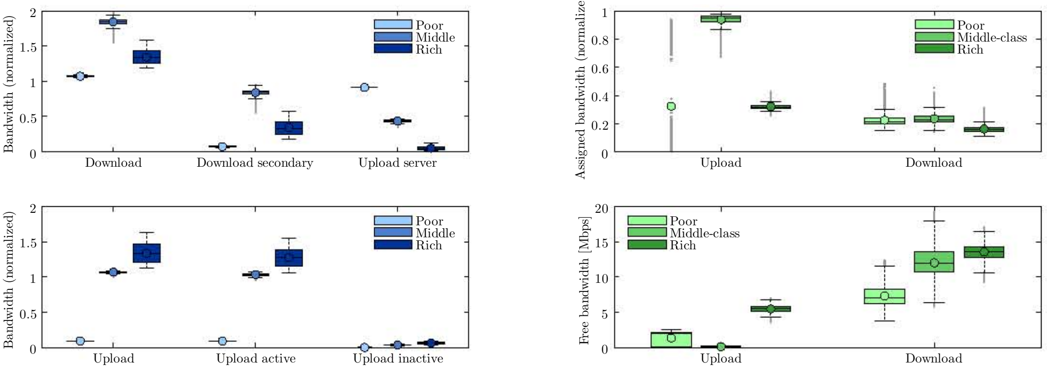
We adopt a conservative policy for the advertisement of all peers as potential uploaders. In order for peers to advertise in any of the PCA peer pools, they must decide whether they have sufficient bandwidth to support an additional stream upload/download, while reserving sufficient bandwidth for the primary streams. When TV channels of different bandwidth requirements are provided in the same package, the peers' bandwidth reservation is done with respect to the channel with the largest number of streams. Therefore, peers may not advertise their resources in the anticipation of moving from an SDTV to an HDTV channel some time in the future.

16.3.1 Peer Participation

To understand the impact of heterogenous TV channels on the P2PTV service performance, we examine the peer participation based on the poor, middle-class, and rich bandwidth scenarios. To this end, the figure 16.8(a) illustrates the session bandwidth, normalized to that of the primary streams. When comparing this result with the homogenous case from the chapter 14, we note the following key differences.

The participation of the poor or middle-class peers is greatly reduced, due to the requirement of reserving their download bandwidth for the use of HDTV channels. The consequence is the increased server contribution, up to 100 % for poor peers, to compensate for the lack of available peer bandwidth. Because peers maintain the bandwidth reservation, despite downloading high-definition channels only a fraction of the time, increases the average free bandwidth, as shown in the figure 16.8(b). The middle-class and rich scenarios are less susceptible, given the high-end peer capabilities.

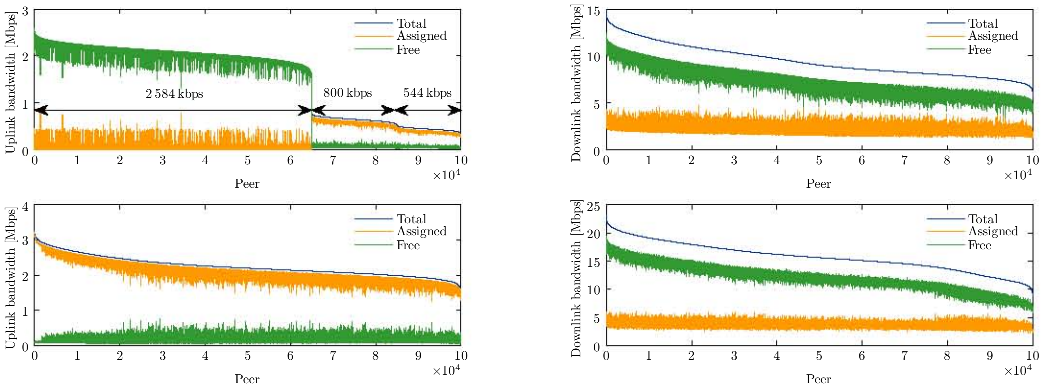
Given the conservative approach to bandwidth reservation, there are fewer differences in terms of the impact on the end-user experience, such as the channel connection delay or churn, and for the sake of simplicity we do not include them here. However, it is worth noting that the impact



(a) Bandwidth assignment, depending on the session type.

(b) Assigned and free peer bandwidth.

Figure 16.8: Performance analysis of the peer participation.



(a) Upload bandwidth: poor (top), middle-class (bottom).

(b) Download bandwidth: poor (top), middle-class (bottom.

Figure 16.9: The distribution of bandwidth bandwidth.

on the service quality may be significant when using the flexible bandwidth reservation, which normally does not provision for high definition channels.

16.3.2 Resource Performance

At last, we examine the distribution of peer resources, which underlines one of the fundamental differences when streaming heterogenous channels. As in the chapter 15, the figures 16.9(a) and 16.9(b) illustrates the total, assigned and free bandwidth for the set of UE peers. Unlike before, the conservation of free bandwidth is no longer similar across the peer set, because some peers are prevented from uploading by not meeting the HDTV bandwidth threshold, even when they would have enough free bandwidth to do so.

Again, the impact is highly pronounced for UE peers poor in resources. In this particular case, the HDTV threshold of 10 Mbps for 5 % of the channels, just above the free downlink rate of a large fraction of peers in upload bandwidth, prevents these peers from properly using their upload resources most of the time, effectively aggravating the resource-poor situation. Although it is not our objective to extend the work further in the direction of heterogenous channels, possible solutions include:

- the centrist approach to bandwidth reservation, as a tradeoff between resource usage and user experience;
- restricting the access to HDTV channels for rich peers, even if the scenario is poor, and;
- using hybrid streaming.

16.4 Conclusions

This chapter extended the experimental evaluation of the P2PTV service to cover a set of questions that were not addressed previously. Among these was the justification that peer-to-peer represents a good choice to classic streaming approaches, such as IP multicast. Since our initial results correctly identify the difficulty in using peer-to-peer alone in bandwidth-poor circumstances, we envision a hybrid streaming alternative where P2PTV complements IP multicast for a subset of streams.

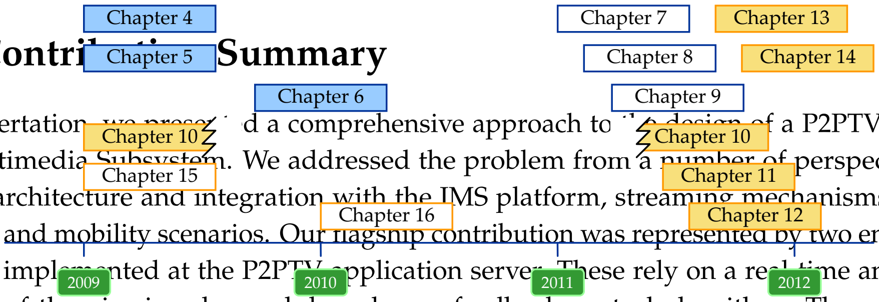
Without entering into greater details, the division between peer-to-peer (or unicast) and multicast may be done at channel level or stream level. The former facilitates the division of content based on popularity, while the latter is useful for scalable coding algorithms, where the base layer uses IP multicast while the enhancement layers peer-to-peer. By relying on both simulation and analytical modeling, we show that P2PTV is particularly useful for unpopular TV channels. Given the long-tail shape of the popularity distribution, such an approach would have only a small cost in terms of bandwidth. On the other hand, the gain in terms of scalability and multicast addresses may be substantial, especially when there are a large number of TV channels or inter-operability between different transport providers is a major issue.

The second question focused on the service performance for heterogenous TV channels. The design characteristic which plays a crucial role in the performance of our P2PTV service, is the ability to anticipate the viewer demand, and commit the necessary network resources in a proactive fashion. When the TV channels require a different number of streams, depending on the configuration the service may over-reserve (conservative) or under-reserve (flexible) the peer bandwidth with the corresponding consequences. Our experimental results show that the peer-to-peer streaming is only affected significantly under resource poor conditions.

Chapter 17

Summary and Future Enhancements

17.1 Contribution Summary



In this dissertation, we presented a comprehensive approach to the design of a P2PTV service for the IP Multimedia Subsystem. We addressed the problem from a number of perspectives, such as service architecture and integration with the IMS platform, streaming mechanisms, signaling procedures and mobility scenarios. Our flagship contribution was represented by two enhancement techniques implemented at the P2PTV application server. These rely on a real-time and adaptive estimation of the viewing demand, based on a feedback-control algorithm. They allow us to determine the number of inactive uploading session for faster signaling procedures, as well as the necessary peer participation and bandwidth assignment for a lower peer churn.

In the interest of simplifying our work, we introduced the enhancement, session and peer coordination, algorithms incrementally. We believe that this step-by-step approach, taking separately each sub-problem, increases the accessibility for the reader. At the same time, given the proof-of-concept level of our contribution, our experimental evaluation focused on scalability and reproducibility of results, rather than a high level of detail.

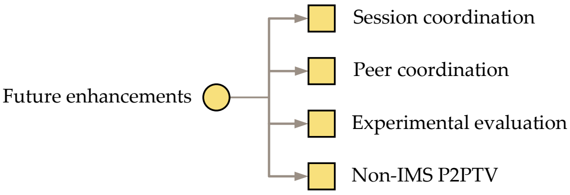
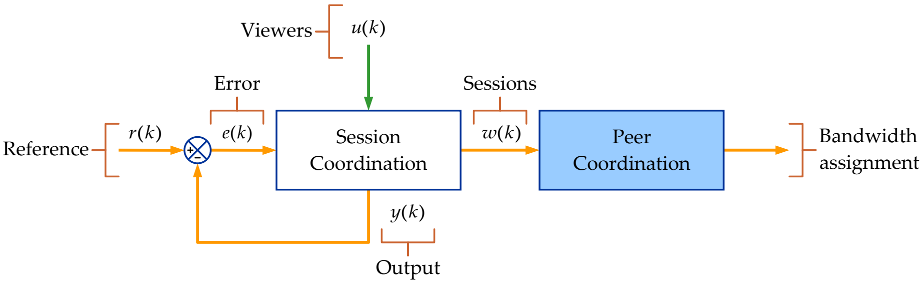
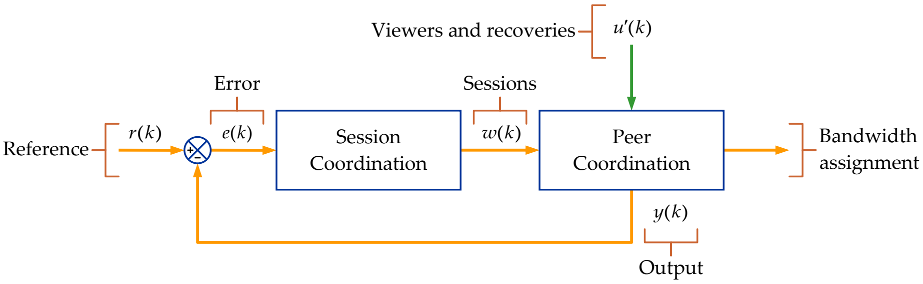


Figure 17.1: Several ideas on future enhancements.

By considering these issues, as well as the lessons learned in the course of our research, we recognize the possibility for future improvements. Toward this goal, the remainder of this chapter sketches several coarse ideas that one may use to further improve this scientific work, summarized



(a) Feedback loop for session coordination.



(b) Feedback loop for session and peer coordination.

Figure 17.2: Enhancing the session coordination.

in the figure 17.1. At last, we complete this dissertation with a summary of our contributions.

17.2 Enhancing the Session Coordination

The session coordination algorithm estimates the viewing demand on every TV channel, by computing the minimum number of uploading sessions that may accommodate the current active users with a selected blocking ratio β. Simultaneously, the session minimization problem is equivalent to maximizing the bandwidth utilization, ρ.

To accomplish this, the control output represents a quantity dependent on the number of new channel users, $u(k)$. For instance, the fast feedback loop measures the difference $w_f(t) - u(k)$, whereas the stochastic and slow outputs are nonlinear functions of the same variables. Ultimately, the number of sessions $w(k)$, computed independently in this fashion by the session coordination algorithm, will reflect only the changes of $u(k)$, as illustrated in the figure 17.2(a).

The drawback of this approach, as we mentioned in the chapter 14, is that peer orphaning caused by churn is not included in the session coordination. Therefore, a session recovery may only benefit from fast signaling at the expense of a higher blocking ratio for new channel arrivals. As indicated at one point, the `PolicyRecoverInactive` allows the P2PTV provider to fine tune this behavior depending on whether the UE peers have poor or rich connections.

While this policy represents an acceptable compromise, it may be possible to use the feedback control loop to compute the number of inactive sessions for both user arrivals and departures. The main distinction is that, whereas a user arrival requires one inactive session, a departure may need zero or several inactive sessions, depending on how many neighbors has the user equipment leaving the network. For this reason the state of the peer coordination algorithm should be included in the control loop, as shown the figure 17.2(b).

In principle, such an approach would provide superior blocking ratio performance with the cost of a lower utilization. The open questions to examine are the structure of the feedback system, and whether there exists a stable transfer function given the churn dynamics. In such a system the peer bandwidth distribution would likely play a key role, in addition to the user activity, and would require special consideration during the design and the analysis of the new system.

17.3 Enhancing the Peer Coordination

In P2PTV, a new user equipment selects one or more source peers that leverage their upstream connection to forward the video content, reducing the costs for the service provider. In this setting, supporting multiple TV channels brings additional challenges, where the users tendency to browse through the available channels increases the peer churn negatively affecting the overall performance (C. Wu et al., 2008). To this end, the view-upload decoupling (VUD) and our peer coordination algorithm (PCA) focus on using additional download and upload bandwidth disconnecting the user activity from the peer uploading. If during the peer selection multiple user equipments meet the uploading conditions, we use the resource level metric to discriminate between candidates.

However, a possible alternative is to enhance the peer selection to estimate their reliability on the current TV channels (F. Wang et al., 2008; Liu et al., 2009; Kermarrec et al., 2009). This approach recognizes that allocating more bandwidth does not always solve the problem, especially when it is a scarce resource. Compared to the current work, which uses heuristic approaches based on intuitive observations, such a solution could be founded on both a probabilistic analysis using the reliability theory and a historical profile of peer participation.

The peer reliability may have a temporal dimension, depending on the online age of the user equipment. For example, if X is a random variable representing the session length and $h_X(x)$ is the corresponding probability density function, the probability of the future lifetime, Y, depending on the elapsed time e, which we can write as $\mathscr{E} = (X > e)$, is:

$$h_Y(y|\mathscr{E}) = \frac{h_X(y+e)}{1 - H_X(e)}. \tag{17.1}$$

The survival probability of an uploading peer, denoted by P_u, equals the probability that its future lifetime Y_u is greater than the future lifetime of its downloading neighbor, Y_d: $P_u = \mathrm{P}\{Y_u \geq Y_d\} = \mathrm{P}\{Y_u - Y_d \geq 0\}$. The probability density function of the difference between two independent random variables is:

$$h_Z(z|\mathscr{E}_u\mathscr{E}_d) = \int_{-\infty}^{\infty} f_Y(x+z|\mathscr{E}_u) f_Y(x|\mathscr{E}_d)\mathrm{d}x, \tag{17.2}$$

where $\mathscr{E}_u$ and $\mathscr{E}_d$ are the elapsed time conditions for the uploading and downloading peers. Then, the survival probability is:

$$P_s = \int_0^{\infty} f_Z(z|\mathscr{E}_u\mathscr{E}_d)\mathrm{d}z. \tag{17.3}$$

17.4 Extending the Experimental Evaluation

The performance evaluation from the chapters 15 and 16 relied on large-scale computer simulations to identify the impact of our proposal on the user experience, in terms of session setup delay and streaming interruptions. At this stage, we identified the simulation environment as a reasonable compromise, especially when we considered the large-scale setup of special importance. It was our intention to consider the audio/video encoding as an abstraction layer, and not to include a particular standard.

However, real-life performance, which is ultimately the quality of the playback at the subscriber's TV set, is sensitive to all these details. Therefore, it would be desirable to extend our evaluation toward enhancing the realism at the following levels:

- the *network level*, by considering packet-based flows, queuing and congestion;
- the *signaling level*, with improved models or real implementation of the IMS call service control functions;
- the *streaming level*, including codec characteristics and packetization of the audio and video data, and;
- the *user level*, by assessing the playback and interaction quality through the prism of a mean opinion score.

17.5 Non-IMS P2PTV

While this work dedicate exclusively to the design of a P2PTV service for the IMS, it is possible that many of the concepts incubated here may be transferred and apply in non-IMS scenarios. Certainly, the QoS reservation and the signaling requirements are a trademark of the IMS multimedia sessions, and it would make little sense considering them outside such a platform. However, using an adaptive feedback-based algorithm to estimate the viewer demand, and leveraging this information to drive the peer participation may find an opportunity in the Internet peer-to-peer streaming.

Glossary

3GPP	3rd Generation Partnership Project
AAA	Authentication, authorization and accounting
ACAA	Active channel-aware algorithm
ADSL	Asymmetric Digital Subscriber Line
ALM	Application-level multicast
AN	Access network
API	Application programming interface
AS	Application Server
AVC	Advanced Video Coding
B2BUA	Back-to-back user agent
BIBO	Bounded input bounded output
BS	Broadcast Server
BSCF	Broadcast Server Communication Function
CCDF	Complementary cumulative distribution function
CDF	Cumulative distribution function
CDN	Content delivery network
CN	Core network
CoA	Care-of address
CP	Correspondent peer
CPA	Correspondent peer address
CPU	Central processing unit
CSCF	Call Session Control Function

DCCP	Datagram Congestion Control Protocol
DHT	Distributed hash table
DFT	Discrete Fourier transform
DLTI	Discrete, linear and time-invariant
DNS	Domain Name System
DSL	Digital Subscriber Line
DVMRP	Distance Vector Multicast Routing Protocol
E-CDF	Empirical cumulative distribution function
ETSI	European Telecommunications Standards Institute
FC	Filter criteria
FQDN	Fully qualified domain name
GGSN	Gateway GPRS Support Node
GPRS	General Packet Radio Service
GW	Gateway
HA	Home agent
HAA	Home agent address
HDTV	High definition television
HoA	Home address
HSS	Home Subscriber Server
I-CSCF	Interrogating Call Session Control Function
IETF	Internet Engineering Task Force
iFC	Initial filter criteria
IGMP	Internet Group Management Protocol
IMPI	IP Multimedia Private Identity
IMPU	IP Multimedia Public Identity
IMS	IP Multimedia Subsystem
IMSU	IP Multimedia Subscription
IP	Internet Protocol
IPTV	Internet Protocol television
LAN	Local area network
LTE	Long term evolution
MCF	Media Control Function
MDF	Media Delivery Function
MF	Media Function
MIH	Media Independent Handover

MIHF	Media Independent Handover Function
MIH-SAP	Media Independent Handover Service Access Point
MIIS	MIH Information Service
MIP	Mobile IP
NAL	Network abstraction layer
NBA	Naïve bandwidth approach
NGN	Next-generation network
NVoD	Near video-on-demand
O-UA	Originating user agent
OSI	Open Systems Interconnection
P2P	Peer-to-peer
P2P-IMS-TV	Peer-to-peer television for the IP Multimedia Subsystem
P2PTV	Peer-to-peer television
P2PTV-AS	Peer-to-peer television Application Server
PCA	Peer coordination algorithm
PCAA	Passive channel-aware algorithm
PCEF	Policy and Charging Enforcement Function
PCRF	Policy and Charging Rules Function
P-CSCF	Proxy Call Session Control Function
PCTS	Proactive Context Transfer Service
PDF	Probability density function
PDP	Packet Data Protocol
PI	Proportional-integral
PID	Proportional-integral-derivative
PIM	Protocol Independent Multicast
PoS	Point-of-service
PSI	Public Service Identity
QoS	Quality of service
RACS	Resource and Admission Control Subsystem
RGW	Residential gateway
RTCP	RTP Control Protocol
RTP	Real-time Transport Protocol
SCA	Session coordination algorithm
SCF	Service Control Function
S-CSCF	Serving Call Session Control Function

SDF	Service Discovery Function
SDP	Session Description Protocol
SDTV	Standard definition television
sFC	Subsequent filter criteria
SGSN	Serving GPRS Support Node
SIP	Session Initiation Protocol
SISO	Single input single output
SLA	Service-level agreement
SM	Session manager
SNR	Signal-to-noise ratio
SPT	Service point trigger
SSF	Service Selection Function
STB	Set-top box
SVC	Scalable Video Coding
TCP	Transmission Control Protocol
TISPAN	Telecommunications and Internet converged Services and Protocols for Advanced Networking
TP	Trigger point
T-UA	Terminating user agent
TV	Television
UA	User agent
UAC	User agent client
UAS	User agent server
UDP	User Datagram Protocol
UE	User Equipment
UMTS	Universal Mobile Telecommunication System
UPSF	User Profile Server Function
URI	Uniform resource identifier
UTRAN	Universal Terrestrial Radio Access Network
VoD	Video on-demand
VoIP	Voice over Internet Protocol
VUD	View-upload decoupling
WLAN	Wireless local area network
XML	eXtensible Markup Language

Notations

Notation	Description
B	Performance evaluation: used network bandwidth
B_A	Performance evaluation: used access network bandwidth
$B_{A,d}$	Performance evaluation: used downlink access network bandwidth
$B_{A,u}$	Performance evaluation: used uplink access network bandwidth
B_C	Performance evaluation: used core network bandwidth
$B_{C,m}$	Performance evaluation: used multicast core network bandwidth
$B_{C,u}$	Performance evaluation: used unicast core network bandwidth
b	Session coordination algorithm: blocking ratio
b_d	Peer coordination algorithm: peer download bandwidth
b_s	Peer coordination algorithm: stream bandwidth
b_u	Peer coordination algorithm: peer upload bandwidth
D	Performance evaluation: delay
D_p	Performance evaluation: processing delay
D_q	Performance evaluation: queueing delay
D_t	Performance evaluation: transmission delay
d_{ch}	User activity: channel session duration
d_{off}	User activity: offline session duration
d_{on}	User activity: online session duration
e_f	Session coordination algorithm: control error of the fast algorithm
e_h	Session coordination algorithm: control error of the stochastic algorithm
e_{ch}	User activity: error between trace and model channel session rate

Notation	Description
e_{off}	User activity: error between trace and model offline session rate
e_{on}	User activity: error between trace and model online session rate
e_s	Session coordination algorithm: control error of the slow algorithm
$\mathcal{E}_{ch}$	User activity: channel events set
$\mathcal{E}_{off}$	User activity: offline events set
$\mathcal{E}_{on}$	User activity: online events set
$\mathcal{I}_{ch}$	User activity: channel intervals set
$\mathcal{I}_{off}$	User activity: offline intervals set
$\mathcal{I}_{on}$	User activity: online intervals set
k	Core network topology: multicast scaling factor according to Chuang and Sirbu law
l_m	Core network topology: average size of the multicast tree
l_u	Core network topology: average distance between edge routers
m	Core network topology: the routers to links ratio
N	Number of TV channels
N'	Number of TV channels with zipf distributed mean popularity
n_a	Peer coordination algorithm: peer number of uploading active streams
n_d	Peer coordination algorithm: peer number of downloading streams
n_i	Peer coordination algorithm: peer number of uploading inactive streams
n_p	Peer coordination algorithm: peer number of downloading primary streams
n_s	Peer coordination algorithm: peer number of downloading secondary streams
n_u	Peer coordination algorithm: peer number of uploading streams
$\mathcal{O}_{ch}$	User activity: channel open events set
$\mathcal{O}_{off}$	User activity: offline open events set
$\mathcal{O}_{on}$	User activity: online open events set
P_t	Channel popularity: instantaneous component
P_μ	Channel popularity: mean popularity
p	Channel popularity: instantaneous popularity
$\mathcal{P}$	Peer coordination algorithm: peer set of non-uploading primary streams
$\mathcal{P}^*$	Peer coordination algorithm: peer set of uploading primary streams
q_b	Channel popularity: backward browsing sequential switching
q_c	Channel popularity: browsing sequential switching
q_f	Channel popularity: forward browsing sequential switching
q_s	Channel popularity: sequential switching
q_t	Channel popularity: target switching

Notation	Description
r	Peer coordination algorithm: peer resource level
r_b	Session coordination algorithm: reference (desired) blocking ratio
r_ρ	Session coordination algorithm: reference utilization
r_{off}	User activity: ratio of offline user equipments
r_{on}	User activity: ratio of online user equipments
r^*_{off}	User activity: modulated ratio of offline user equipments
r^*_{on}	User activity: modulated ratio of online user equipments
$\mathcal{R}$	Peer coordination algorithm: peer resource set
$\mathcal{S}$	Peer coordination algorithm: peer set of secondary streams
U	Number of TV subscribers
u	Number of online TV subscribers
u_{off}	Number of offline TV subscribers
u_{on}	Number of online TV subscribers
w	Session coordination algorithm: number of upload sessions
w_a	Session coordination algorithm: number of active upload sessions
w_f	Session coordination algorithm: control input of the fast algorithm
w_h	Session coordination algorithm: control input of the stochastic algorithm
w_i	Session coordination algorithm: number of inactive upload sessions
w_s	Session coordination algorithm: control input of the slow algorithm
X_{ch}	User activity: random normalized channel session rate
X_{off}	User activity: random normalized offline session rate
X_{on}	User activity: random normalized online session rate
X^*_{off}	User activity: modulated random and normalized offline session rate
X^*_{on}	User activity: modulated random and normalized online session rate
x_{ch}	User activity: mean normalized channel session rate
x_{off}	User activity: mean normalized offline session rate
x_{on}	User activity: mean normalized online session rate
x^*_{off}	User activity: modulated mean and normalized offline session rate
x^*_{on}	User activity: modulated mean and normalized online session rate
y_f	Session coordination algorithm: control output of the fast algorithm
y_s	Session coordination algorithm: control output of the slow algorithm
z_{ch}	User activity: mean absolute channel session rate
z_{off}	User activity: mean absolute offline session rate
z_{on}	User activity: mean absolute online session rate

Notation	Description
z^*_{off}	User activity: modulated mean and absolute offline session rate
z^*_{on}	User activity: modulated mean and absolute online session rate
β	Session coordination algorithm: reference (desired) blocking ratio
ϵ_{off}	User activity: workload synthesis offline session error
ϵ_{on}	User activity: workload synthesis online session error
ρ	Session coordination algorithm: utilization
σ	Channel popularity: random permutation of a TV channel

References

3GPP. (2007). *IP Multimedia Subsystem (IMS); Stage 2.* 3GPP/TISPAN TS 23.517 version 8.0.0 Release 8.

3GPP. (2010). *Open Service Access (OSA).* 3GPP TS 23.198 version 9.0.0 Release 9.

3GPP. (2011a). *3G security; Network Domain Security (NDS); IP network layer security.* 3GPP TS 33.210 version 10.3.0 Release 10.

3GPP. (2011b). *Customised Applications for Mobile network Enhanced Logic (CAMEL) Phase 4.* 3GPP TS 23.278 version 10.0.0 Release 10.

3GPP. (2011c). *Telecommunication management; Charging management; Charging architecture and principles.* 3GPP TS 32.240 version 10.1.0 Release 10.

3GPP. (2012a). *IP multimedia call control protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3.* 3GPP TS 24.229 version 10.6.1 Release 10.

3GPP. (2012b). *IP Multimedia (IM) Subsystem Cx and Dx Interfaces; Signalling flows and message contents.* 3GPP TS 29.228 version 10.4.0 Release 10.

3GPP. (2012c). *IP Multimedia Subsystem (IMS); Stage 2.* 3GPP TS 23.228 version 10.7.0 Release 10.

3GPP. (2012d). *Multimedia Resource Function Controller (MRFC); Multimedia Resource Function Processor (MRFP); Mp interface: Procedures descriptions.* 3GPP TS 23.333 version 10.3.0 Release 10.

3GPP. (2012e). *Numbering, addressing and identification.* 3GPP TS 23.003 version 10.4.0 Release 10.

3GPP. (2012f). *Policy and charging control architecture.* 3GPP TS 23.203 version 10.6.0 Release 10.

3GPP. (2012g). *Policy and charging control over Gx/Sd reference point.* 3GPP TS 29.212 version 10.5.0 Release 10.

3GPP. (2012h). *Policy and charging control over Rx reference point.* 3GPP TS 29.214 version 10.5.0 Release 10.

3GPP. (2012i). *Policy and charging control signalling flows and Quality of Service (QoS) parameter mapping.* 3GPP TS 29.213 version 10.4.0 Release 10.

Aboba, B., & Beadles, M. (1999). *The Network Access Identifier.* RFC 2486.

Ali, S., Mathur, A., & Zhang, H. (2006). Measurement of commercial peer-to-peer live video streaming. In *Proceedings of workshop in recent advances in peer-to-peer streaming.*

Bikfalvi, A., García-Reinoso, J., Vidal, I., & Valera, F. (2009a, April). Nozzilla: A Peer-to-Peer IPTV Distribution Service for an IMS-based NGN. In *Proceedings of the 5th international conference on networking and services (iaria icns 2009)* (pp. 450–455). IEEE Computer Society.

Bikfalvi, A., García-Reinoso, J., Vidal, I., & Valera, F. (2009b, June–July). A peer-to-peer IPTV service architecture for the IP multimedia subsystem. *International Journal of Communication Systems*, *23*(6–7), 780–801.

Bikfalvi, A., García-Reinoso, J., Vidal, I., Valera, F., & Azcorra, A. (2011, April). P2P vs. IP multicast: comparing approaches to IPTV streaming based on TV channel popularity. *Computer Networks*, *55*(6), 1310–1325.

Camarillo, G., & Garcia-Martin, M. (2011). *The 3G IP multimedia subsystem (IMS): merging the Internet and the cellular worlds.* Wiley.

Camarillo, G., Marshall, W., & Rosenberg, J. (2002). Integration of resource management and Session Initiation Protocol (SIP). *Integration.* RFC 3312 (Proposed Standard).

Castro, M., Druschel, P., Kermarrec, A., Nandi, A., Rowstron, A., & Singh, A. (2003). SplitStream: high-bandwidth multicast in cooperative environments. In *Acm sigops operating systems review* (pp. 298–313).

Castro, M., Druschel, P., Kermarrec, A., & Rowstron, A. (2002). SCRIBE: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications*, *20*(8), 1489–1499.

Cha, M., Rodriguez, P., Crowcroft, J., Moon, S., & Amatriain, X. (2008). Watching Television Over an IP Network. In *Proceedings of the 8th acm sigcomm conference on internet measurement (imc 2008)* (pp. 71–84).

Cha, M., Rodriguez, P., Moon, S., & Crowcroft, J. (2008). On next-generation telco-managed P2P TV architectures. In *Proceedings of the 7th international conference on peer-to-peer systems* (pp. 5–5).

Chiba, T., Yokota, H., Dutta, A., Chee, D., & Schulzrinne, H. (2008). Performance analysis of next-generation mobility protocols for IMS/MMD networks. In *Proceedings of the wireless communications and mobile computing conference (iwcmc 2008)* (pp. 68–73).

Chu, Y., Rao, S., & Zhang, H. (2000). A case for end system multicast. In *Acm sigmetrics performance evaluation review* (Vol. 28, pp. 1–12).

Chuang, J., & Sirbu, M. (2001). Pricing multicast communication: A cost-based approach. *Telecommunication Systems*, *17*(3), 281–297.

Cisco. (2012a). Forecast and Methodology, 2011–2016. *Visual Networking Index.*

Cisco. (2012b). The Zettabyte Era. *Visual Networking Index.*

Cunningham, G., Perry, P., Murphy, J., & Murphy, L. (2009). Seamless handover of IPTV streams in a wireless LAN network. *IEEE Transactions on Broadcasting*, *55*(4), 796–801.

D'Agostino, R., Belanger, A., & D'Agostino Jr, R. (1990). A suggestion for using powerful and informative tests of normality. *American Statistician*, 316–321.

D'Agostino, R., & Pearson, E. (1973). Tests for departure from normality. Empirical results for the distributions of b_2 and $\sqrt{b_1}$. *Biometrika*, *60*(3), 613–622.

Deering, S. E. (1986). *Host Extensions for IP Multicasting.* RFC 988.

de la Oliva, A., Banchs, A., Soto, I., Melia, T., & Vidal, A. (2008). An overview of IEEE 802.21: media-independent handover services. *IEEE Wireless Communications*, *15*(4), 96–103.

Devarapalli, V., Wakikawa, R., Petrescu, A., & Thubert, P. (2005). *Network mobility (nemo) basic support protocol.* RFC 3963 (Proposed Standard).

Dixit, A., Pindyck, R., & Davis, G. (1994). *Investment under uncertainty* (Vol. 15). Princeton University Press Princeton, NJ.

Dutta, A., Manousakis, K., Das, S., & Lin, F. (2007). Mobility testbed for 3GPP2-based multimedia domain networks. *IEEE Communications Magazine*, *45*(7), 118–126.

Eriksson, H. (1994). The multicast backbone. *Communications of the ACM*, *8*, 54–60.

Fei, A., Cui, J., Gerla, M., & Faloutsos, M. (2001). Aggregated Multicast: an approach to reduce multicast state. In *Proceedings of the ieee global telecommunications conference (globecom 2001)* (Vol. 3, pp. 1595–1599).

Feldmann, A., & Whitt, W. (1998). Fitting mixtures of exponentials to long-tail distributions to analyze network performance models. *Performance evaluation*, *31*(3-4), 245–279.

Gupta, V. (2008). *IEEE802.21 Standard and Metropolitan Area Networks: Media Independent Handover Services.*

Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., & Stewart, L. (1999). *HTTP Authentication: Basic and Digest Access Authentication.* RFC 2617.

Handley, M., Jacobson, V., & Perkins, C. (2006). *Sdp: Session description protocol.* RFC 4566 (Proposed Standard).

Hei, X., Liang, C., Liang, J., Liu, Y., & Ross, K. (2007). A measurement study of a large-scale P2P IPTV system. *IEEE Transactions on Multimedia*, *9*(8), 1672–1687.

Hei, X., Liu, Y., & Ross, K. (2008). IPTV over P2P streaming networks: the mesh-pull approach. *IEEE Communications Magazine*, *46*(2), 86–92.

Heine, G., & Sagkob, H. (2003). *GPRS: gateway to third generation mobile networks*. Artech House Publishers.

Hellerstein, J., Diao, Y., Parekh, S., & Tilbury, D. (2004). *Feedback control of computing systems.* Wiley Online Library.

ITU-T. (2011). *Advanced video coding for generic audiovisual services.* ITU-T Recommendation H.264 (06/11).

Jannotti, J., Gifford, D., Johnson, K., Kaashoek, M., & O'Toole, J. (2000). Overcast: reliable multicasting with on overlay network. In *Proceedings of the 4th conference on symposium on operating system design & implementation* (Vol. 4, pp. 14–14).

Jewell, N. (1982). Mixtures of exponential distributions. *The Annals of Statistics*, 479–484.

Johnson, D., Perkins, C., & Arkko, J. (2004). *Mobility support in IPv6.* RFC 3775 (Proposed Standard).

Kermarrec, A., Le Merrer, E., Liu, Y., & Simon, G. (2009). Surfing peer-to-peer IPTV: distributed channel switching. *Euro-Par 2009 Parallel Processing*, 574–586.

Kloeden, P., Platen, E., & Schurz, H. (1994). *Numerical solution of SDE through computer experiments* (Vol. 1). Springer Verlag.

Koodli, R. (2009). *Mobile IPv6 fast handovers.* RFC 5568 (Proposed Standard).

Lee, C., & Knight, D. (2005). Realization of the next-generation network. *IEEE Communications Magazine*, *43*(10), 34–41.

Liu, Z., Wu, C., Li, B., & Zhao, S. (2009). Distilling superior peers in large-scale P2P streaming systems. In *Proceedings of the 28th ieee international conference on computer communications (infocom 2009)* (pp. 82–90).

Loughney, J., Guttman, E., Zorn, E., & Arkko, J. (2003). *Diameter Base Protocol.* RFC 3588 (Standard).

Magharei, N., & Rejaie, R. (2009). PRIME: Peer-to-peer receiver-driven mesh-based streaming. *IEEE/ACM Transactions on Networking*, *17*(4), 1052–1065.

Magharei, N., Rejaie, R., & Guo, Y. (2007). Mesh or multiple-tree: A comparative study of live P2P streaming approaches. In *Proceedings of the 26th ieee international conference on computer communications (infocom 2007)* (pp. 1424–1432).

Más, I., Berggren, V., Jana, R., Murray, J., & Rice, C. (2008). IMS-TV: An IMS-based architecture for interactive, personalized IPTV. *IEEE Communications Magazine*, *46*(11), 156–163.

Maymounkov, P., & Mazieres, D. (2002). Kademlia: A peer-to-peer information system based on the XOR metric. *Peer-to-Peer Systems*, 53–65.

Mikoczy, E., Sivchenko, D., Xu, B., & Rakocevic, V. (2007). IMS based IPTV services: architecture and implementation. In *Proceedings of the 3rd international conference on mobile multimedia communications* (p. 15).

Munir, A. (2008). Analysis of SIP-based IMS session establishment signaling for WiMax-3G networks. In *Proceedings of the 4th international conference on networking and services (icns 2008)* (pp. 282–287).

Perkins, C. (2002). *IP mobility support for IPv4.* RFC 3344 (Proposed Standard).

Perkins, C., & Westerlund, M. (2010). *Multiplexing RTP Data and Control Packets on a Single Port.* RFC 5761 (Proposed Standard).

Pesch, D., Pous, M., & Foster, G. (2005). Performance evaluation of SIP-based multimedia services in UMTS. *Computer Networks*, *49*(3), 385–403.

Petrie, D., & Channabasappa, S. (2011). *A Framework for Session Initiation Protocol User Agent Profile Delivery.* RFC 6080 (Proposed Standard).

Phillips, G., Shenker, S., & Tangmunarunkit, H. (1999). Scaling of multicast trees: Comments on the chuang-sirbu scaling law. In *Acm sigcomm computer communication review* (Vol. 29, pp. 41–51).

Qiu, T., Ge, Z., Lee, S., Wang, J., Xu, J., & Zhao, Q. (2009). Modeling user activities in a large IPTV system. In *Proceedings of the 9th acm sigcomm conference on internet measurement (imc 2009)* (pp. 430–441).

Qiu, T., Ge, Z., Lee, S., Wang, J., Zhao, Q., & Xu, J. (2009). Modeling channel popularity dynamics in a large IPTV system. In *Proceedings of the 11th international joint conference on measurement and modeling of computer systems (acm sigmetrics 2009)* (pp. 275–286).

Radoslavov, P., Estrin, D., & Govindan, R. (1999). *Exploiting the bandwidth-memory tradeoff in multicast state aggregation* (Tech. Rep. No. 99-697 (Second Revision)). Information Sciences Institute, 4676 Admiralty Way, Suite 1001, Marina Del Rey, CA 90292-6695, USA: Department of Computer Science, University of Southern California.

Ratnasamy, S., Francis, P., Handley, M., Karp, R., & Shenker, S. (2001). A scalable content-addressable network. *ACM SIGCOMM Computer Communication Review*, *31*(4), 161–172.

Renier, T., Larsen, K., Castro, G., & Schwefel, H. (2007). Mid-session macro-mobility in IMS-based networks. *IEEE Vehicular Technology Magazine*, *2*(1), 20–27.

Roach, A. B. (2002). *Session Initiation Protocol (SIP)-Specific Event Notification.* RFC 3265.

Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., … Schooler, E. (2002). *SIP: Session Initiation Protocol.* RFC 3261.

Rowstron, A., & Druschel, P. (2001). Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware 2001* (pp. 329–350).

Schulzrinne, H., & Casner, S. (2003). *RTP profile for Audio and Video Conferences with Minimal Control.* RFC 3551 (Standard).

Schulzrinne, H., Casner, S., Frederick, R., & Jacobson, V. (2003). *RTP: A Transport Protocol for Real-Time Applications.* RFC 3550 (Standard).

Shapiro, S., & Wilk, M. (1965). An analysis of variance test for normality. *Biometrika*, *52*(3/4), 591–611.

Silverston, T., & Fourmaux, O. (2007). Measuring P2P IPTV systems. In *Proceedings of the 17th sigmm workshop on network and operating systems support for digital audio and video (acm nossdav 2007)* (Vol. 7).

Soliman, H., ElMalki, K., Bellier, L., & Castelluccia, C. (2008). *Hierarchical mobile IPv6 (HMIPv6) mobility management.* RFC 5380 (Proposed Standard).

Sripanidkulchai, K., Ganjam, A., Maggs, B., & Zhang, H. (2004). The feasibility of supporting large-scale live streaming applications with dynamic application end-points. In *Acm sigcomm computer communication review* (Vol. 34, pp. 107–120).

Stoica, I., Morris, R., Karger, D., Kaashoek, M., & Balakrishnan, H. (2001). Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM Computer Communication Review*, *31*(4), 149–160.

Tang, W., Fu, Y., Cherkasova, L., & Vahdat, A. (2003). Medisyn: A synthetic streaming media service workload generator. In *Proceedings of the 13th international workshop on network and operating systems support for digital audio and video* (pp. 12–21).

Thaler, D., & Handley, M. (2000). On the aggregatability of multicast forwarding state. In *Proceedings of the 19th ieee international conference on computer communications (infocom 2000)* (Vol. 3, pp. 1654–1663).

TISPAN. (2008). *NGN Functional Architecture.* TISPAN TS 82.001 version 3.4.1 Release 3.

TISPAN. (2011a). *IPTV Architecture; IPTV functions supported by the IMS subsystem.* TISPAN TS 82.027 version 3.5.1 Release 3.

TISPAN. (2011b). *Resource and Admission Control Subsystem.* TISPAN ES 82.003 version 3.5.1 Release 3.

Ulvan, M., & Bestak, R. (2009). Analysis of Session Establishment Signaling Delay in IP Multimedia Subsystem. *Wireless and Mobile Networking*, 44–55.

Venkataraman, V., Yoshida, K., & Francis, P. (2006). Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast. In *Proceedings of the 14th ieee international conference on network protocols (icnp 2006)* (pp. 2–11).

Vidal, I., García-Reinoso, J., de la Oliva, A., Bikfalvi, A., & Soto, I. (2010, September). Supporting mobility in an IMS-based P2P IPTV service: A proactive context transfer mechanism. *Computer Communications (Special Issue on Multimedia Networking and Security in Convergent Networking)*, *33*(14), 1736–1751.

Vidal, I., García-Reinoso, J., Valera, F., & Bikfalvi, A. (2009, November). Enabling layered video coding for IMS-based IPTV home services. *IEEE Network*, *23*(6), 30–35.

Vlavianos, A., Iliofotou, M., & Faloutsos, M. (2006). BiToS: Enhancing BitTorrent for supporting streaming applications. In *Proceedings of the 25th ieee international conference on computer communications (infocom 2006)* (pp. 1–6).

Vu, L., Gupta, I., Liang, J., & Nahrstedt, K. (2007). Measurement of a large-scale overlay for multimedia streaming. In *Proceedings of the 16th international symposium on high performance distributed computing* (pp. 241–242).

Wang, F., Liu, J., & Xiong, Y. (2008). Stable peers: Existence, importance, and application in peer-to-peer live video streaming. In *Proceedings of the 27th ieee international conference on computer communications (infocom 2008)* (pp. 1364–1372).

Wang, M., Xu, L., & Ramamurthy, B. (2010). Linear programming models for multi-channel P2P streaming systems. In *Proceedings of the 29th ieee international conference on computer communications (infocom 2010)* (pp. 1–5).

Wang, Y. K., Kristensen, T., & Jesup, R. (2011). *RTP Payload Format for H.264 Video.* RFC 6184 (Proposed Standard).

Wang, Y. K., Schierl, T., & Eleftheriadis, A. (2011). *RTP Payload Format for Scalable Video Coding.* RFC 6190 (Proposed Standard).

Westerlund, M. (2004). *A Transport Independent Bandwidth Modifier for the Session Description Protocol (SDP).* RFC 3890 (Proposed Standard).

Wiegand, T., Sullivan, G., Bjontegaard, G., & Luthra, A. (2003). Overview of the H. 264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, *13*(7), 560–576.

Wong, T., & Katz, R. (2000). An analysis of multicast forwarding state scalability. In *Proceedings of the 8th ieee international conference on network protocols (icnp 2000)* (pp. 105–115).

Wu, C., Li, B., & Zhao, S. (2008). Multi-channel live P2P streaming: Refocusing on servers. In *Proceedings of the 27th ieee international conference on computer communications (infocom 2008)* (pp. 1355–1363).

Wu, D., Liang, C., Liu, Y., & Ross, K. (2009). View-upload decoupling: A redesign of multi-channel P2P video systems. In *Proceedings of the 28th ieee international conference on computer communications (infocom 2009)* (pp. 2726–2730).

Wu, D., Liu, Y., & Ross, K. (2009). Queuing network models for multi-channel P2P live streaming systems. In *Proceedings of the 28th ieee international conference on computer communications (infocom 2009)* (pp. 73–81).

Xia, F., Sarikaya, B., & Patil, B. (2008). *Mobile ipv6 handover using home agent buffering.* Internet Draft (draft-xia-mipshop-ha-buffering-01).

Xiao, Y., Du, X., Zhang, J., Hu, F., & Guizani, S. (2007). Internet protocol television (IPTV): the killer application for the next-generation internet. *IEEE Communications Magazine*, *45*(11), 126–134.

Xie, S., Keung, G., & Li, B. (2007). A measurement of a large-scale peer-to-peer live video streaming system. In *Proceedings of the international conference on parallel processing workshops (icpp 2007)* (p. 57).

Yu, G., Westholm, T., Kihl, M., Sedano, I., Aurelius, A., Lagerstedt, C., & Odling, P. (2009). Analysis and characterization of IPTV user behavior. In *Proceedings of the ieee international symposium on broadband multimedia systems and broadcasting (bmsb 2009)* (pp. 1–6).

Zhang, B., & Mouftah, H. (2003). Forwarding state scalability for multicast provisioning in IP networks. *IEEE Communications Magazine*, *41*(6), 46–51.

Zhang, M., Zhang, Q., Sun, L., & Yang, S. (2007). Understanding the power of pull-based streaming protocol: Can we do better? *IEEE Journal on Selected Areas in Communications*, *25*(9), 1678–1694.

Zhang, X., Liu, J., Li, B., & Yum, Y. (2005). CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming. In *Proceedings of the 24th ieee international conference on computer communications (infocom 2005)* (Vol. 3, pp. 2102–2111).

Zhao, B., Lui, J., & Chiu, D. (2009). Exploring the optimal chunk selection policy for data-driven P2P streaming systems. In *Proceedings of the 9th ieee international conference on peer-to-peer computing (p2p 2009)* (pp. 271–280).

Part

VI

Appendices

Appendix A

Signaling Procedures for the P2PTV Service

A.1 Overview

The chapter 5 presented the IMS signaling that is the most relevant to our contribution. However, in the interest of keeping the description simple, we have omitted some details regarding the signaling steps between network entities. In this appendix, we present in full the signaling procedures for the P2PTV service. Their purpose is to complement the content of the chapter 5 for the advanced reader.

The figure A.1 summarizes all procedures used by the P2PTV service. We classify them according to whether they are initiated by the UE or by the P2PTV-AS due to an external event, such as an action by the user (turning on/off the TV set-top box or changing the channel), or an action by the AS service logic (software event generated by the SCA or PCA). We exclude from our list the procedures that are required for operation of the service, but do not focus explicitly on the P2PTV operations, including the registration, de-registration, subscription to other event packages, etcetera.

As mandated by the IMS standards, we assume that all UE multimedia sessions require the use of preconditions before the acceptance of the session and the initiation of the media stream. These include the mechanisms for bearer establishment explained in the TS 23.228 (3GPP, 2012c) and SDP precondition options from RFC 3312 (Camarillo et al., 2002). As a trusted entity in the core network, the broadcast server(s) do not require preconditions for the associated multimedia sessions.

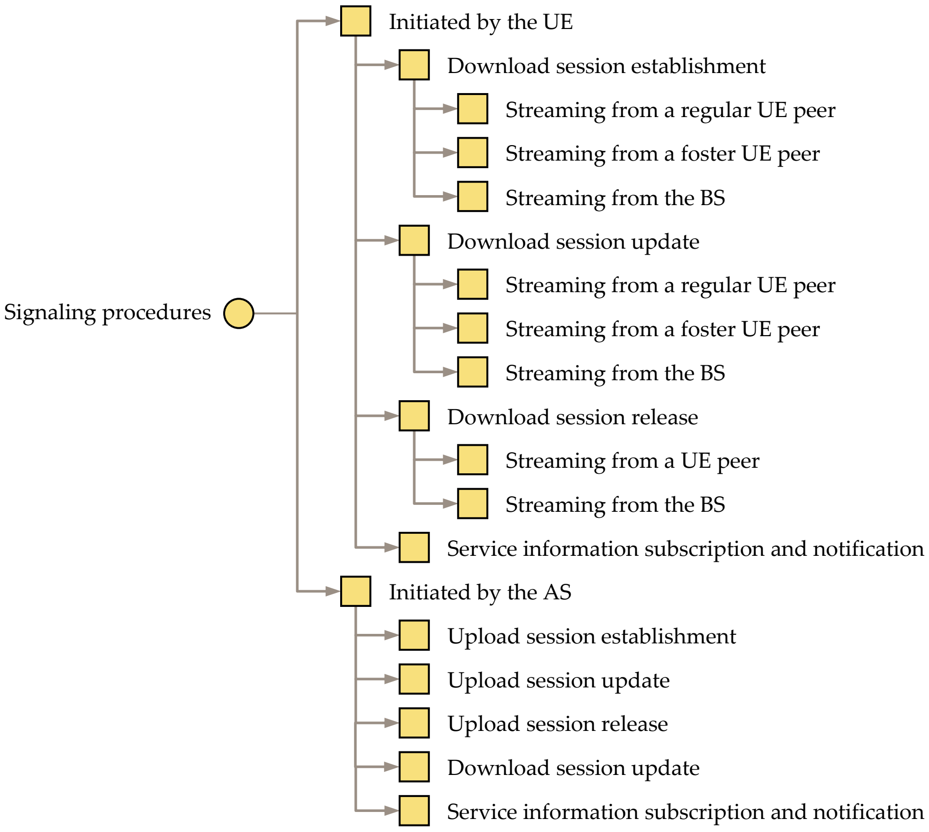

Signaling procedures
Initiated by the UE
Download session establishment
Streaming from a regular UE peer
Streaming from a foster UE peer
Streaming from the BS
Download session update
Streaming from a regular UE peer
Streaming from a foster UE peer
Streaming from the BS
Download session release
Streaming from a UE peer
Streaming from the BS
Service information subscription and notification
Initiated by the AS
Upload session establishment
Upload session update
Upload session release
Download session update
Service information subscription and notification

Figure A.1: The signaling procedures related to the operation of the P2PTV service.

A.2 Procedures Initiated by the User Equipment

The user equipment controls the multimedia sessions corresponding to the downloaded streams. Depending on the type of user interaction, we have the following procedures:

- session *establishment*, when the user turns on the UE;
- session *update*, when the user changes the current TV channel, and;
- session *release*, when the user turns off the UE.

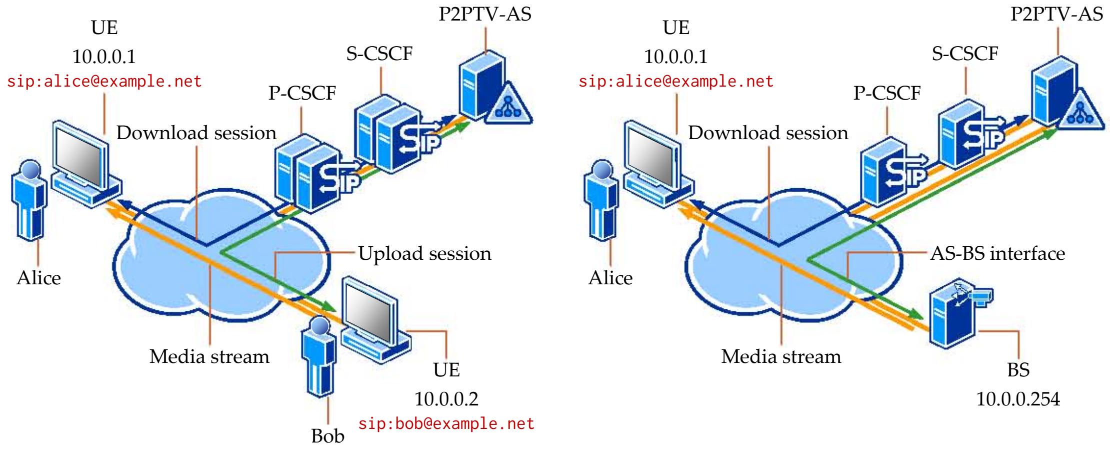
In addition to multimedia sessions, the UE controls the establishment and the release of the subscription dialog to the P2PTV service notifications.

The signaling of the streaming sessions depends on the type of the uploading entity, which may be:

- a *regular UE peer*, without an established and inactive upload session;
- a *foster UE peer*, with an established and inactive upload session, or;
- the *broadcast server*.

The first two cases correspond to the scenario illustrated in the figure A.2(a), where the UE of the subscriber Alice, having the public user identity `sip:alice@example.net` and the unicast IP address `10.0.0.1` downloads the video stream from the UE of the subscriber Bob, with the public user identity `sip:bob@example.net` and the unicast IP address `10.0.0.2`.

The figure A.2(b) shows the situation when the UE downloads the video stream from the broadcast server, having the unicast IP address `10.0.0.3`. Although the specification of the interface



(a) Peer-to-peer streaming.

(b) Broadcast server streaming.

Figure A.2: The streaming scenarios used for the signaling procedures.

between the P2PTV-AS and the BS is outside the scope of our work, within this appendix we shall assume that the required protocol is SIP, and the signaling conforms with a third-party multimedia session without preconditions as mandated by the section 5.7a.2 of the TS 23.228 (3GPP, 2012c)[1]. The signaling communication between the P2PTV-AS may be direct, if the P2PTV-AS has SIP routing capabilities, or may be performed via the IMS core network.

A.2.1 Download Session Establishment

A.2.1.1 Streaming From a Regular UE Peer

During the establishment of a multimedia session for a given video stream, the peer coordination algorithm of the P2PTV-AS may select a regular UE peer without an inactive upload session. This situation occurs when the P2PTV-AS is configured with a higher priority for P2P streaming rather than BS streaming for new users. The figure A.3 illustrates the steps of the corresponding signaling procedure.

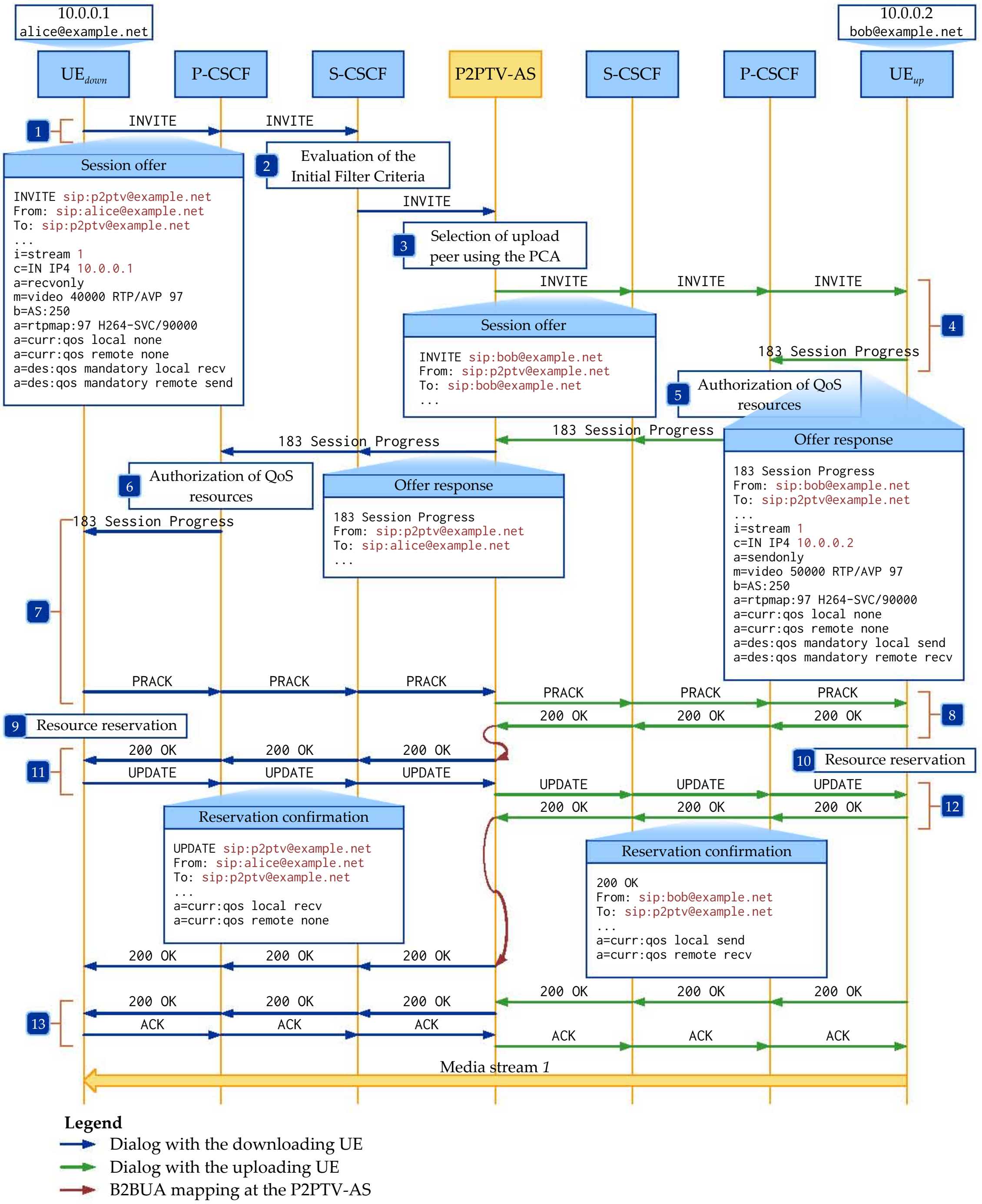
The procedure for the establishment of a download session, with a regular UE as uploading peer, is the following.

1. The downloading UE sends an `INVITE` SIP request to the associated P-CSCF, known by the UE from the initial P-CSCF discovery. The origin of the SIP message is the public user identity of the current subscriber (`sip:alice@example.net`), and the destination is the public service identity of the P2PTV service (`sip:p2ptv@example.net`). The request contains an SDP body describing the session offer, formatted according to the rules from the section A.4. The `INVITE` message must contain a `Route` header containing the URI of the S-CSCF that was assigned during registration. Upon receiving the `INVITE` request, the P-CSCF forwards the message to the S-CSCF.

2. The S-CSCF performs the service control by validating the message against the initial filter criteria corresponding to the user's service profile. By considering the service profile described in the section B, the S-CSCF forwards the request to P2PTV-AS.

3. The P2PTV-AS uses the peer coordination algorithm to service the request, that is determining a network entity (a UE or the BS) that can upload the requested stream. Under the current

[1] At present the IPTV specification of TISPAN is in progress, and the *y*2 reference point between the IMS core network and the media control function (MCF) is undefined.



Session offer

```
INVITE sip:p2ptv@example.net
From: sip:alice@example.net
To: sip:p2ptv@example.net
...
i=stream 1
c=IN IP4 10.0.0.1
a=recvonly
m=video 40000 RTP/AVP 97
b=AS:250
a=rtpmap:97 H264-SVC/90000
a=curr:qos local none
a=curr:qos remote none
a=des:qos mandatory local recv
a=des:qos mandatory remote send
```

Session offer

```
INVITE sip:bob@example.net
From: sip:p2ptv@example.net
To: sip:bob@example.net
...
```

Offer response

```
183 Session Progress
From: sip:bob@example.net
To: sip:p2ptv@example.net
...
i=stream 1
c=IN IP4 10.0.0.2
a=sendonly
m=video 50000 RTP/AVP 97
b=AS:250
a=rtpmap:97 H264-SVC/90000
a=curr:qos local none
a=curr:qos remote none
a=des:qos mandatory local send
a=des:qos mandatory remote recv
```

Offer response

```
183 Session Progress
From: sip:p2ptv@example.net
To: sip:alice@example.net
...
```

Reservation confirmation

```
UPDATE sip:p2ptv@example.net
From: sip:alice@example.net
To: sip:p2ptv@example.net
...
a=curr:qos local recv
a=curr:qos remote none
```

Reservation confirmation

```
200 OK
From: sip:bob@example.net
To: sip:p2ptv@example.net
...
a=curr:qos local send
a=curr:qos remote recv
```

Figure A.3: The session establishment procedure when the stream is uploaded by a regular peer.

scenario, the P2PTV-AS creates a SIP B2BUA instance, which generates a new SIP dialog and sends an `INVITE` request to the public user identity (`sip:bob@example.net`) of the uploading UE. The request contains a sequence of `Route` headers describing the route to the destination UE[2]. The request contains an SDP with the same session information as in the initial request sent by the Alice's UE.

4. Upon receiving the `INVITE` request, the Bob's UE verifies that the uploading conditions are met (the requested stream is available, the network conditions allow the upload, etc.), and replies with a `183 Session Progress` response containing the SDP response to the initial offer.

5. The P-CSCF assigned to Bob's UE uses the SDP information from the `183 Session Progress` reply to authorize the required QoS resources. The figure does not illustrate the signaling steps corresponding to this authorization, which conform to the procedure explained in the TS 29.214 (3GPP, 2012h). The P-CSCF forwards the reply message to P2PTV-AS via Bob's S-CSCF. The P2PTV-AS received the `183 Session Progress` reply at the SIP B2BUA instance mapping the downloading and uploading sessions, and it replies with a `183 Session Progress` message on the dialog with Alice, containing the same session description. The P2PTV-AS sends the reply via Alice's S-CSCF and P-CSCF, respectively.

6. Upon receiving the offer response, Alice's P-CSCF authorizes the QoS resources for the requested session, and forwards the reply to the UE.

7. The Alice's UE confirms the response, by sending a provisional acknowledge or `PRACK` request to Bob's UE via the B2BUA instance in the P2PTV-AS.

8. The Bob's UE acknowledges the confirmation with a `200 OK` response.

9. Upon sending the `PRACK` request, the Alice's UE initiates the resource reservation procedure, according to its current access network.

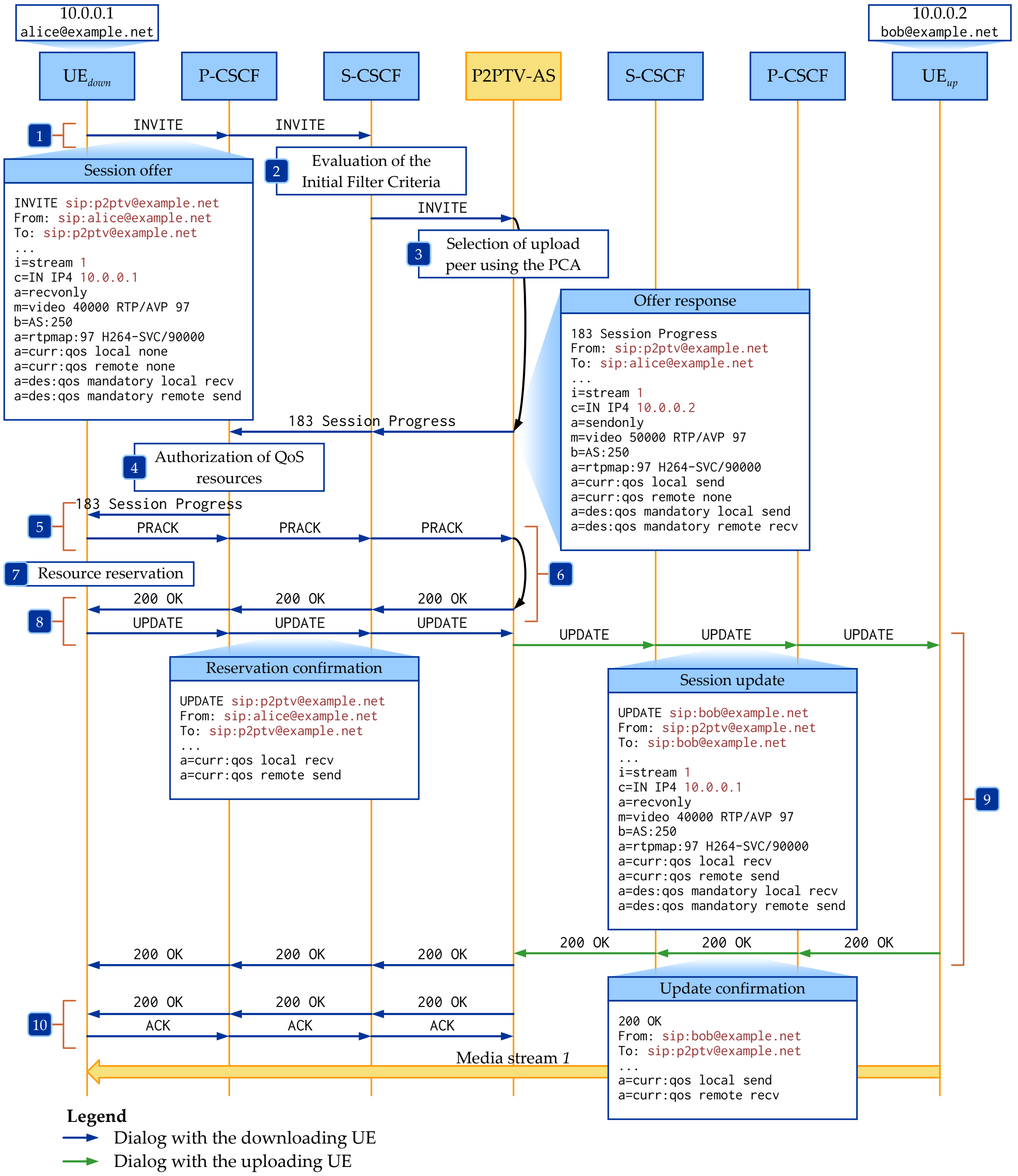
10. Upon sending the `200 OK` response, the Bob's UE initiates the resource reservation procedure, according to its current access network.

11. When the Alice's UE completes the resource reservation, it sends a reservation confirmation `UPDATE` request to Bob's UE via the P2PTV-AS. The request contains an SDP body describing the updated QoS reservation.

12. When the Bob's UE completes the resource reservation, and upon receiving the `UPDATE` request, it replies with a `200 OK` response. The response contains an SDP body describing the updated QoS reservation.

13. Finally, the Bob's UE completes the session establishment procedure by replying with a `200 OK` response to the initial `INVITE` request, via the P2PTV-AS. The Alice's UE acknowledges the reply with an `ACK` sent to Bob's UE via the P2PTV-AS.

Following the completion of the signaling procedure at each end, the Bob's UE begins uploading the requested stream and the Alice's UE begins processing the stream at its decoding layer.



Session offer

```
INVITE sip:p2ptv@example.net
From: sip:alice@example.net
To: sip:p2ptv@example.net
...
i=stream 1
c=IN IP4 10.0.0.1
a=recvonly
m=video 40000 RTP/AVP 97
b=AS:250
a=rtpmap:97 H264-SVC/90000
a=curr:qos local none
a=curr:qos remote none
a=des:qos mandatory local recv
a=des:qos mandatory remote send
```

Offer response

```
183 Session Progress
From: sip:p2ptv@example.net
To: sip:alice@example.net
...
i=stream 1
c=IN IP4 10.0.0.2
a=sendonly
m=video 50000 RTP/AVP 97
b=AS:250
a=rtpmap:97 H264-SVC/90000
a=curr:qos local send
a=curr:qos remote none
a=des:qos mandatory local send
a=des:qos mandatory remote recv
```

Reservation confirmation

```
UPDATE sip:p2ptv@example.net
From: sip:alice@example.net
To: sip:p2ptv@example.net
...
a=curr:qos local recv
a=curr:qos remote send
```

Session update

```
UPDATE sip:bob@example.net
From: sip:p2ptv@example.net
To: sip:bob@example.net
...
i=stream 1
c=IN IP4 10.0.0.1
a=recvonly
m=video 40000 RTP/AVP 97
b=AS:250
a=rtpmap:97 H264-SVC/90000
a=curr:qos local recv
a=curr:qos remote send
a=des:qos mandatory local recv
a=des:qos mandatory remote send
```

Update confirmation

```
200 OK
From: sip:bob@example.net
To: sip:p2ptv@example.net
...
a=curr:qos local send
a=curr:qos remote recv
```

Figure A.4: The session establishment procedure when the stream is uploaded by a foster peer.

A.2.1.2 Streaming From a Foster UE Peer

When the uploading UE is a foster peer, the signaling procedure on the downloading side is similar to the situation where the uploading peer is a regular peer. However, on the uploading side, the P2PTV-AS uses the existing inactive session. The figure A.4 illustrates the steps involved, which are the following.

1. The same as the step 1 from the procedure A.2.1.1.
2. The same as the step 2 from the procedure A.2.1.1.
3. The P2PTV-AS uses the peer coordination algorithm to service the request. Under the current scenario, the P2PTV-AS assigns an existing foster peer and creates a SIP B2BUA instance mapping the originating dialog to the inactive upload session of the foster peer. By using the session information stored from the inactive dialog, the P2PTV-AS replies to the initial `INVITE` with a `183 Session Progress` response, which contains the session information as if would have been generated by Bob's UE.
4. The same as the step 6 from the procedure A.2.1.1.
5. The same as the step 7 from the procedure A.2.1.1.
6. Upon receiving the provisional acknowledge, the P2PTV-AS replies with a `200 OK` to Alice's UE.
7. The same as the step 9 from the procedure A.2.1.1.
8. The same as the step 11 from the procedure A.2.1.1.
9. The same as the step 12 from the procedure A.2.1.1.
10. The same as the step 13 from the procedure A.2.1.1.

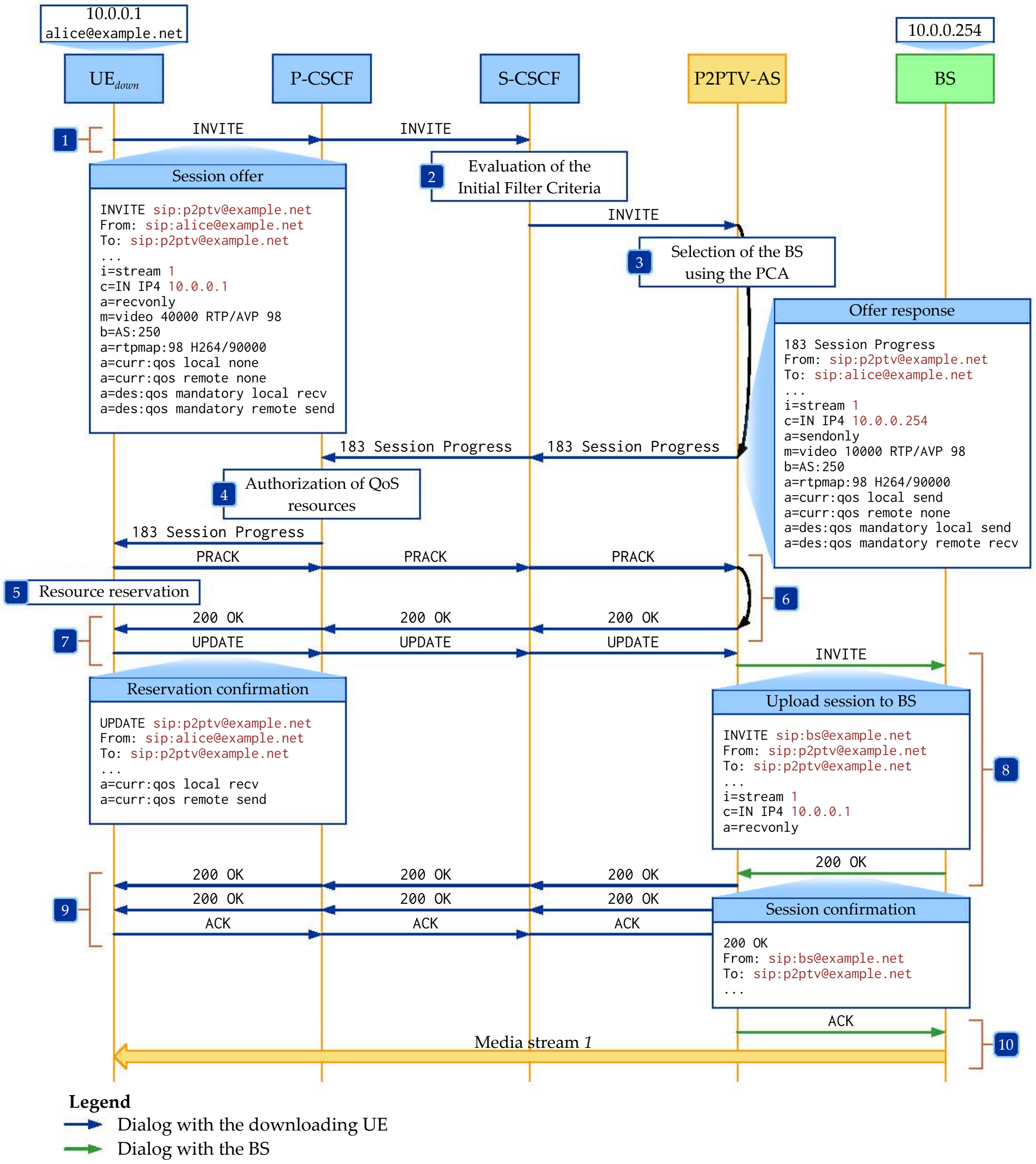
Following the completion of the signaling procedure at each end, the Bob's UE begins uploading the requested stream and the Alice's UE begins processing the stream at its decoding layer.

A.2.1.3 Streaming From the Broadcast Server

As mentioned previously, when the P2PTV-AS uses the BS during a session establishment, we assume that the exchange between the P2PTV-AS and the BS is identical to a SIP session establishment without preconditions. In addition, the P2PTV-AS is configured, either statically by the operator or with an out-of-band exchange with the BS, with the associated connection information. The figure A.5 illustrates the steps involved, which are the following.

1. The same as the step 1 from the procedure A.2.1.1.
2. The same as the step 2 from the procedure A.2.1.1.
3. The P2PTV-AS uses the peer coordination algorithm to service the request. Under the current scenario, a suitable uploading peer cannot be found, and the P2PTV-AS assigns the BS as an uploading entity. Upon receiving the `INVITE` request, the P2PTV-AS replies to Alice's UE

[2] The SIP routing data associated to a given UE is learned by the P2PTV-AS from the service subscription dialog, and is recorded in the peer information record. Recording to route to the UE peers prevents the P2PTV-AS from forwarding the initial requests to a different S-CSCF and/or the I-CSCF serving the home network of the uploading UE. However, it requires the UE to update the routing information when the current network, P-CSCF or S-CSCF change. In addition, it requires the S-CSCF to recognize and authorize the public service identity of the P2PTV service. Whenever these constraints cannot be met, the P2PTV-AS may not include any `Route` headers, and may forward the request to its default S-CSCF.



Session offer

```
INVITE sip:p2ptv@example.net
From: sip:alice@example.net
To: sip:p2ptv@example.net
...
i=stream 1
c=IN IP4 10.0.0.1
a=recvonly
m=video 40000 RTP/AVP 98
b=AS:250
a=rtpmap:98 H264/90000
a=curr:qos local none
a=curr:qos remote none
a=des:qos mandatory local recv
a=des:qos mandatory remote send
```

Offer response

```
183 Session Progress
From: sip:p2ptv@example.net
To: sip:alice@example.net
...
i=stream 1
c=IN IP4 10.0.0.254
a=sendonly
m=video 10000 RTP/AVP 98
b=AS:250
a=rtpmap:98 H264/90000
a=curr:qos local send
a=curr:qos remote none
a=des:qos mandatory local send
a=des:qos mandatory remote recv
```

Reservation confirmation

```
UPDATE sip:p2ptv@example.net
From: sip:alice@example.net
To: sip:p2ptv@example.net
...
a=curr:qos local recv
a=curr:qos remote send
```

Upload session to BS

```
INVITE sip:bs@example.net
From: sip:p2ptv@example.net
To: sip:p2ptv@example.net
...
i=stream 1
c=IN IP4 10.0.0.1
a=recvonly
```

Session confirmation

```
200 OK
From: sip:bs@example.net
To: sip:p2ptv@example.net
...
```

Figure A.5: The session establishment procedure when the stream is uploaded by the broadcast server.

with a `183 Session Progress` response, containing an SDP body with the associated session information. This information includes the IP address and port of the BS. Depending on the service configuration, these parameters must either be available to the P2PTV-AS through a pre-established mechanism, or the resource reservation at the UE should not be impacted by their actual values, i.e. the UE access network must not filter media packets based on the source IP address and/or transport port. In the latter situation, the P2PTV-AS may use a set of dummy connection and media parameters.

4. The same as the step 6 from the procedure A.2.1.1.

5. The same as the step 9 from the procedure A.2.1.1.

6. The same as the step 6 from the procedure A.2.1.2.

7. When the Alice's UE completes the resource reservation, it sends a reservation confirmation `UPDATE` request to the P2PTV-AS. Upon receiving the request, the P2PTV-AS creates a new SIP B2BUA instance generating an `INVITE` request sent to the BS. The request contains an SDP body with the session information received from Alice's UE.

8. The BS replies with a `200 OK` response, which contains an SDP body with the corresponding session information.

9. Upon receiving the `200 OK` response, the P2PTV-AS generates a similar response on the UE dialog. Depending on whether the previous reply to the UE contained the correct connection/media parameters in the SDP body, the P2PTV-AS may update this data in the current response. In addition, unless the session precondition attributes are supported by the BS, the P2PTV-AS must include them in the `200 OK` response to the UE. Subsequently, the P2PTV-AS generates a `200 OK` response to the initial `INVITE` request. Finally, the Alice's UE completes the session establishment procedure by acknowledging the reply.

10. A similar `ACK` request is returned by the P2PTV-AS to the BS confirming the session establishment. Following the completion of the signaling procedure, the BS begins uploading the requested stream and the Alice's UE begins processing the stream at its decoding layer.

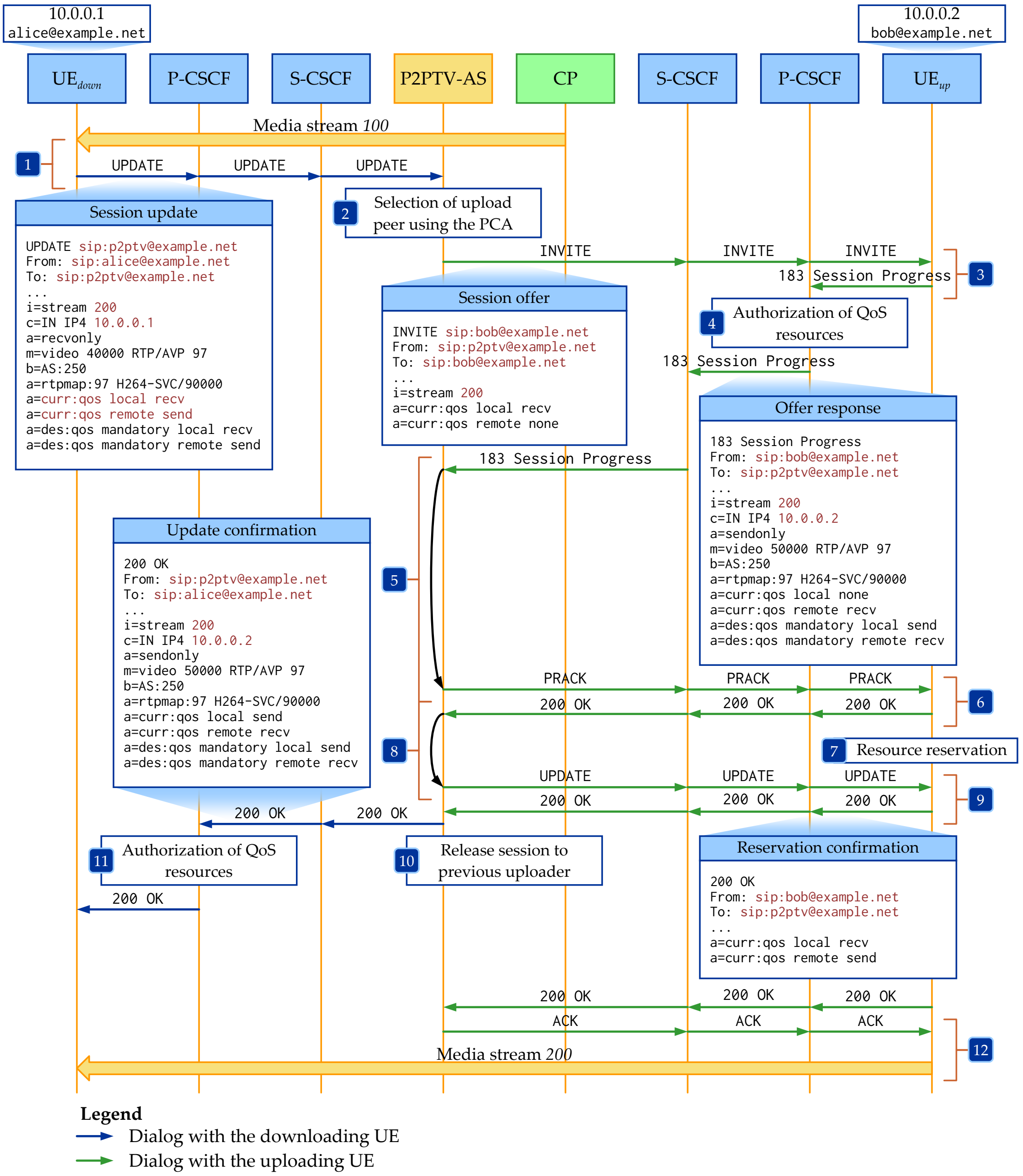
A.2.2 Download Session Update

A.2.2.1 Streaming From a Regular UE Peer

The updating of a download session is initiated by UE during a channel change. In the first scenario, the peer coordination algorithm of the P2PTV-AS may select a regular UE peer without an established inactive upload session. According to the design of the PCA, this situation should be an exception, due to its latency, but it may occur if the configuration of the P2PTV-AS allows it. Therefore, in the interest of completion, we include it in this appendix. The figure A.6 illustrates the steps involved, which are included below.

1. The Alice's UE, currently receiving the stream 100, sends a SIP `UPDATE` request on the dialog of the existing download session, which includes an SDP body describing the session update. The session updates due to channel change will set the session information (`i`) to the new stream 200, and update the connection (`c`) and the media (`m`) fields when necessary. Generally, the session update should not modify the QoS preconditions and bandwidth requirements. The `UPDATE` request is forwarded to the P2PTV-AS.

2. The same as the step 3 from the procedure A.2.1.1, except that the P2PTV-AS uses the same B2BUA instance by removing the previous originating dialog from the mapping, and by adding the new dialog to Bob's UE.



Session update

```
UPDATE sip:p2ptv@example.net
From: sip:alice@example.net
To: sip:p2ptv@example.net
...
i=stream 200
c=IN IP4 10.0.0.1
a=recvonly
m=video 40000 RTP/AVP 97
b=AS:250
a=rtpmap:97 H264-SVC/90000
a=curr:qos local recv
a=curr:qos remote send
a=des:qos mandatory local recv
a=des:qos mandatory remote send
```

Session offer

```
INVITE sip:bob@example.net
From: sip:p2ptv@example.net
To: sip:bob@example.net
...
i=stream 200
a=curr:qos local recv
a=curr:qos remote none
```

Offer response

```
183 Session Progress
From: sip:bob@example.net
To: sip:p2ptv@example.net
...
i=stream 200
c=IN IP4 10.0.0.2
a=sendonly
m=video 50000 RTP/AVP 97
b=AS:250
a=rtpmap:97 H264-SVC/90000
a=curr:qos local none
a=curr:qos remote recv
a=des:qos mandatory local send
a=des:qos mandatory remote recv
```

Update confirmation

```
200 OK
From: sip:p2ptv@example.net
To: sip:alice@example.net
...
i=stream 200
c=IN IP4 10.0.0.2
a=sendonly
m=video 50000 RTP/AVP 97
b=AS:250
a=rtpmap:97 H264-SVC/90000
a=curr:qos local send
a=curr:qos remote recv
a=des:qos mandatory local send
a=des:qos mandatory remote recv
```

Reservation confirmation

```
200 OK
From: sip:bob@example.net
To: sip:p2ptv@example.net
...
a=curr:qos local recv
a=curr:qos remote send
```

Figure A.6: The session update procedure when the new stream is uploaded by a regular peer.

3. The same as the step 4 from the procedure A.2.1.1.

4. The P-CSCF assigned to Bob's UE uses the SDP information from the `183 Session Progress` reply to authorize the required QoS resources, similar to the step 5 from the procedure A.2.1.1. The P-CSCF forwards the reply message to P2PTV-AS via Bob's S-CSCF.

5. Upon receiving the offer response, the P2PTV-AS confirms the response on behalf of Alice's UE by sending a `PRACK` request to Bob's UE.

6. The same as the step 8 from the procedure A.2.1.1.

7. The same as the step 10 from the procedure A.2.1.1.

8. Upon receiving the `200 OK` response, the P2PTV-AS sends an `UPDATE` reservation confirmation back to Bob's UE on behalf of Alice's UE. The request contains an SDP body with the connection information of Alice's UE, as recorded by the P2PTV-AS during the initial `UPDATE` from the step 1.

9. When the Bob's UE completes the resource reservation, and upon receiving the UPDATE request, it replies with a `200 OK` response to the P2PTV-AS. The response contains an SDP body describing the updated QoS reservation. The reply is forwarded by the P2PTV-AS to Alice's UE.

10. Upon receiving the reservation confirmation, the P2PTV-AS changes the session to the previous uploading entity, according to the rules of the PCA. If the previous uploader was another UE peer, the P2PTV-AS may change the upload session to inactive, when an extra foster peer is requires, or may release the session and free the allocated network resources. Otherwise, if the previous uploader was the BS, the P2PTV-AS will release the session.

11. On receiving the `200 OK` response, the Alice's P-CSCF updates the authorization of QoS resources according to the new description information, according to the TS 29.214 (3GPP, 2012h).

12. Finally, the Bob's UE completes the establishment of the upload session by replying with a `200 OK` response to the initial INVITE request from the P2PTV-AS. The P2PTV-AS acknowledges the reply with an `ACK` sent to Bob's UE. Following the completion of the update procedure, the Bob's UE begins uploading the stream 200, and the Alice's UE begins processing the stream at its decoding layer.

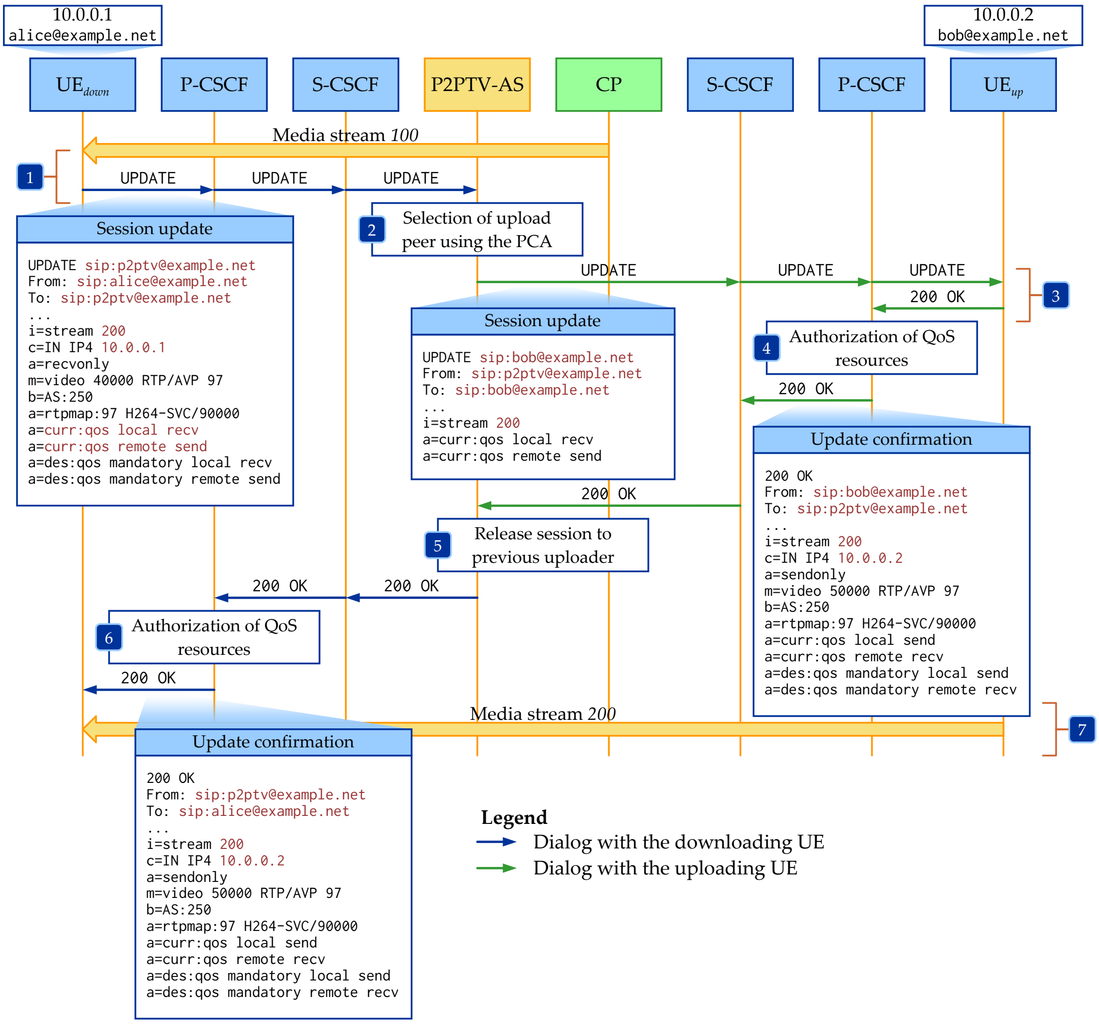
A.2.2.2 Streaming From a Foster UE Peer

The figure A.7 illustrates the signaling procedure for the UE-initiated update of a download session, when the new uploading UE is a foster peer. The steps involved are the following.

1. The same as the step 1 from the procedure A.2.2.1.

2. The same as the step 3 from the procedure A.2.1.1, except that the P2PTV-AS uses the same B2BUA instance by removing the previous originating dialog from the mapping, and by adding an existing inactive dialog to Bob's UE. The P2PTV-AS sends an `UPDATE` request with the request URI set the Bob's public user identity (`sip:bob@example.net`), containing an SDP body with the session description received from Alice's UE. The SDP must not include the `a=inactive` attribute.

3. Upon receiving the UPDATE request, the Bob's UE replies with a `200 OK` response to the P2PTV-AS. The response contains an SDP body with the session information.



Session update

```
UPDATE sip:p2ptv@example.net
From: sip:alice@example.net
To: sip:p2ptv@example.net
...
i=stream 200
c=IN IP4 10.0.0.1
a=recvonly
m=video 40000 RTP/AVP 97
b=AS:250
a=rtpmap:97 H264-SVC/90000
a=curr:qos local recv
a=curr:qos remote send
a=des:qos mandatory local recv
a=des:qos mandatory remote send
```

Session update

```
UPDATE sip:bob@example.net
From: sip:p2ptv@example.net
To: sip:bob@example.net
...
i=stream 200
a=curr:qos local recv
a=curr:qos remote send
```

Update confirmation

```
200 OK
From: sip:bob@example.net
To: sip:p2ptv@example.net
...
i=stream 200
c=IN IP4 10.0.0.2
a=sendonly
m=video 50000 RTP/AVP 97
b=AS:250
a=rtpmap:97 H264-SVC/90000
a=curr:qos local send
a=curr:qos remote recv
a=des:qos mandatory local send
a=des:qos mandatory remote recv
```

Update confirmation

```
200 OK
From: sip:p2ptv@example.net
To: sip:alice@example.net
...
i=stream 200
c=IN IP4 10.0.0.2
a=sendonly
m=video 50000 RTP/AVP 97
b=AS:250
a=rtpmap:97 H264-SVC/90000
a=curr:qos local send
a=curr:qos remote recv
a=des:qos mandatory local send
a=des:qos mandatory remote recv
```

Figure A.7: The session update procedure when the new stream is uploaded by a foster peer.

4. The same as the step 4 from the procedure A.2.2.1.

5. Upon receiving the update confirmation, the P2PTV-AS changes the session to the previous uploading entity, according to the rules of the PCA, similar to the step 10 from the procedure A.2.2.1. The P2PTV-AS confirms the session update to Alice's UE by sending a `200 OK` response with the same session description received from Bob's UE.

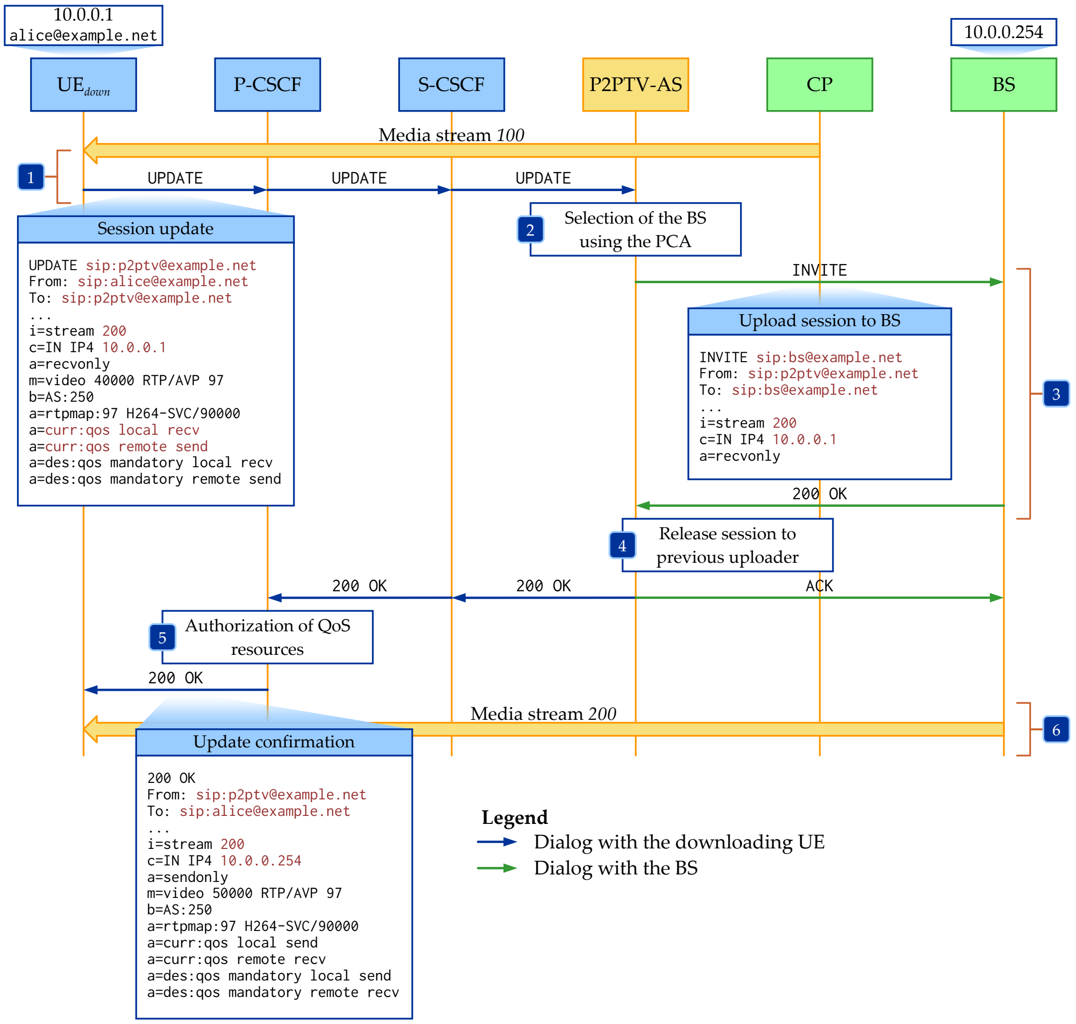
6. The same as the step 11 from the procedure A.2.2.1.

7. Following the completion of the update procedure, the Bob's UE begins uploading the stream 200, and the Alice's UE begins processing the stream at its decoding layer.

A.2.2.3 Streaming From the Broadcast Server

The P2PTV-AS may use the broadcast server during a session update, when there are no regular or foster peer that can upload the requested stream. As in the case of the UE session establishment, we assume that downloading a stream from the BS uses a SIP session without preconditions. The figure A.8 illustrates the steps involved, which are explained below.

1. The same as the step 1 from the procedure A.2.2.1.



Session update

```
UPDATE sip:p2ptv@example.net
From: sip:alice@example.net
To: sip:p2ptv@example.net
...
i=stream 200
c=IN IP4 10.0.0.1
a=recvonly
m=video 40000 RTP/AVP 97
b=AS:250
a=rtpmap:97 H264-SVC/90000
a=curr:qos local recv
a=curr:qos remote send
a=des:qos mandatory local recv
a=des:qos mandatory remote send
```

Upload session to BS

```
INVITE sip:bs@example.net
From: sip:p2ptv@example.net
To: sip:bs@example.net
...
i=stream 200
c=IN IP4 10.0.0.1
a=recvonly
```

Update confirmation

```
200 OK
From: sip:p2ptv@example.net
To: sip:alice@example.net
...
i=stream 200
c=IN IP4 10.0.0.254
a=sendonly
m=video 50000 RTP/AVP 97
b=AS:250
a=rtpmap:97 H264-SVC/90000
a=curr:qos local send
a=curr:qos remote recv
a=des:qos mandatory local send
a=des:qos mandatory remote recv
```

Figure A.8: The session update procedure when the new stream is uploaded by the broadcast server.

2. The P2PTV-AS uses the peer coordination algorithm to service the request. Under the current scenario, a suitable uploading peer cannot be found, and the P2PTV-AS assigns the BS as an uploading entity. Upon receiving the `UPDATE` request, the P2PTV-AS creates a new SIP B2BUA instance generating an `INVITE` request sent to the BS. The request contains an SDP body with the session information received from Alice's UE.

3. The same as the step 8 from the procedure A.2.2.3.

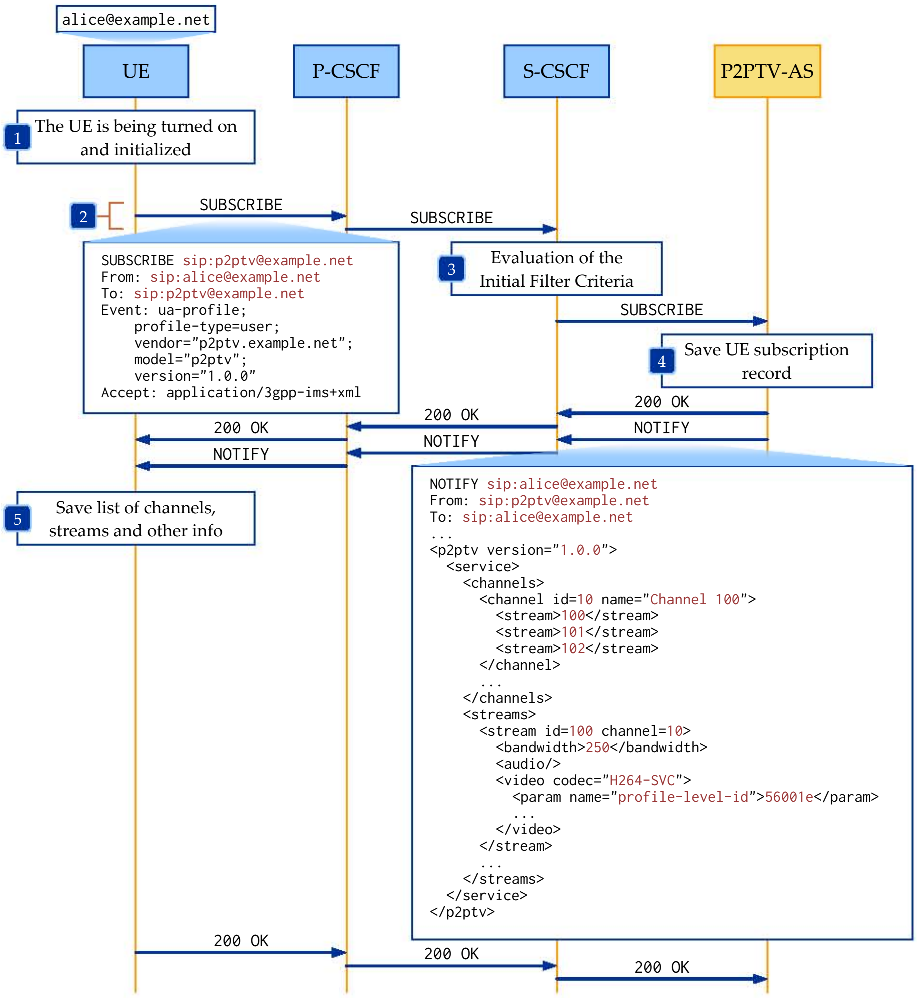
4. The same as the step 5 from the procedure A.2.2.2, except that, in addition, the P2PTV-AS acknowledges the reply from the BS, with an `ACK` request.

5. The same as the step 11 from the procedure A.2.2.1.

6. Following the completion of the update procedure, the BS begins uploading the stream 200, and the Alice's UE begins processing the stream at its decoding layer.

A.2.3 Download Session Release

The user equipment releases a download session in any of the following circumstances:


alice@example.net
UE
P-CSCF
S-CSCF
P2PTV-AS
1
The UE is being turned on and initialized
2
SUBSCRIBE
SUBSCRIBE
3
Evaluation of the Initial Filter Criteria
SUBSCRIBE
4
Save UE subscription record
200 OK
200 OK
200 OK
NOTIFY
NOTIFY
NOTIFY
5
Save list of channels, streams and other info
200 OK
200 OK
200 OK

```
SUBSCRIBE sip:p2ptv@example.net
From: sip:alice@example.net
To: sip:p2ptv@example.net
Event: ua-profile;
    profile-type=user;
    vendor="p2ptv.example.net";
    model="p2ptv";
    version="1.0.0"
Accept: application/3gpp-ims+xml
```

```
NOTIFY sip:alice@example.net
From: sip:p2ptv@example.net
To: sip:alice@example.net
...
<p2ptv version="1.0.0">
  <service>
    <channels>
      <channel id=10 name="Channel 100">
        <stream>100</stream>
        <stream>101</stream>
        <stream>102</stream>
      </channel>
      ...
    </channels>
    <streams>
      <stream id=100 channel=10>
        <bandwidth>250</bandwidth>
        <audio/>
        <video codec="H264-SVC">
          <param name="profile-level-id">56001e</param>
          ...
        </video>
      </stream>
      ...
    </streams>
  </service>
</p2ptv>
```

Figure A.9: The UE subscription to the service notifications.

- the UE is being gracefully turned off by the user, or;
- the UE is downloading a secondary stream without having one or more upload sessions, active or inactive, for the same stream.

The figure 5.3, from the chapter 5, illustrated the signaling steps. We note, that the departure notification, done according to the procedure A.2.5.1, is only necessary when the UE is currently uploading the stream corresponding to the terminating session. Otherwise, when the terminating session is an inactive one, the UE may send a BYE request directly, without a previous notification.

A.2.4 Service Information Subscription

The P2PTV user equipments use the subscription dialog to receive notifications from the P2PTV-AS with the state of the P2PTV service. Currently, our proposal implements only the distribution of channel and stream information during the initialization of the UEs. The subscription uses the *user* profile from the User Agent Profile Delivery event package, as standardized by RFC 6080 (Petrie & Channabasappa, 2011).

The figure A.9 illustrates the setup of the subscription dialog, usually performed immediately after the UE registration. The steps involved are the following.

1. The UE is turned on and registers to the IMS using the Alice's public user identity.

Code sample A.1: The XML body for a session termination notification.

```
<?xml version="1.0" encoding="UTF-8"?>
<ims-3gpp version=1>
  <p2ptv version="1.0.0">
    <action type="bye" data="1000@ue.example.net">
      <stream>1</stream>
    </action>
  </p2ptv>
</ims-3gpp>
```

2. The UE sends a `SUBSCRIBE` request, with the request URI set to the public service identity of the P2PTV service. The subscription SIP message includes an `Event` header set to the `ua-profile` event package, with the following options:

   ```
   profile-type=user;vendor="p2ptv.example.net";model="p2ptv";version="1.0.0"
   ```

   The `SUBSCRIBE` request also includes an `Accept` header set to the value `application/3gpp-ims+xml`, indicating the MIME type bodies that are accepted in notifications.

3. The same as the step 2 from the procedure A.2.1.1.

4. Upon receiving the `SUBSCRIBE` request, the P2PTV-AS saves the UE information and subscription record for subsequent notifications, and it confirms the subscription by sending to the UE: a `200 OK` response, and a `NOTIFY` request including the service information, such as:

   - the available TV channels, and the streams corresponding to each channel, and;
   - the media information for each stream including the bandwidth requirements, the audio and video codecs.

5. The UE saves the service information, and uses it to display the available channels to the user, and to generate the stream session information during channel selection. The UE confirms the notification with a `200 OK` response.

In our figure, we assume the video codec is H.264 Scalable Video Coding. However, the same XML body may be used to specify any alternative codec.

A.2.5 User Equipment Notifications

The UE sends notifications to the P2PTV-AS on the AS subscription to UE state information, setup according to the procedure A.3.5. All notifications involve a `NOTIFY–200 OK` exchange, compliant to RFC 3265 (Roach, 2002), where the notifier is the UE peer, and the subscriber is the P2PTV-AS. All notification requests include an XML body with the MIME type `application/3gpp-ims+xml`, conforming to the schema defined in the section A.5 and the TS 24.229 (3GPP, 2012a).

A.2.5.1 Session Termination

For session termination, the request contains an XML body with at least one `action` element having the `type` attribute set to `bye`, and the `data` attribute set to the call ID of the terminating session. The code sample A.1 represents an example of the XML used for the notification of session release.

Code sample A.2: The XML body for a session failure and request for recovery notification.

```
<?xml version=”1.0” encoding=”UTF-8”?>
<ims-3gpp version=1>
  <p2ptv version=”1.0.0”>
    <action type=”recover” data=”1000@ue.example.net”>
      <stream>1</stream>
    </action>
  </p2ptv>
</ims-3gpp>
```

A.2.5.2 Session Failure

It is not the purpose of our present work to compile a list of all possible failures that may be detected by the UE, and should be reported to the P2PTV-AS. Therefore, the philosophy behind the UE notification is not to report an error, although such a feature may be added in the future, but instead to offer to the P2PTV-AS a corrective action when such a failure occurs. To this end, the UE uses the same XML format as for the session termination, where the UE may include one or more `action` elements.

For example, the code sample A.2 illustrates a possible XML body related to the unexpected failure on receiving the stream 1. In this example, this failure is reported to the P2PTV-AS by including an `action` element, where the `type` attribute is set to `recover`, and the `data` attribute includes the call ID of the corresponding SIP dialog. Such a notification informs the P2PTV-AS that the uploading entity should be updated, due to an unexpected failure.

A.2.5.3 User Equipment Information

Code sample A.3: The XML body for a user equipment information notification.

```
<?xml version=”1.0” encoding=”UTF-8”?>
<ims-3gpp version=1>
  <p2ptv version=”1.0.0”>
    <peer>
      <bwtotal>
        <upload>2584</upload>
        <download>15000</download>
      </bwtotal>
      <bwfree>
        <upload>1834</upload>
        <download>12250</download>
      </bwfree>
    </peer>
  </p2ptv>
</ims-3gpp>
```

The UE uses the P2PTV-AS subscription to send to the P2PTV-AS updated information on the state of the user equipment. In our current work, the UE reports include only the bandwidth available to the UE for the IPTV service, based on the type and status of the current access network. The code sample A.3 shows an example XML body for such a notification.

A.3 Procedures Initiated by the Application Server

The P2PTV application server controls the multimedia sessions corresponding to the uploaded streams. As in the case of the user equipments, we have the following procedures:

- session *establishment*, when the P2PTV-AS allocates a new peer upload session;

- session *update*, used when the upload session changes the destination party, or between active and inactive states, and;
- session *release*, when the P2PTV-AS releases an existing upload session.

On a limited basis[3], the P2PTV-AS may also modify the state of an existing UE download session, when the corresponding peer stops uploading the stream because it is turning off, leaving the service, or is allocating its upload/download resources to other TV channels. When such an event occurs, the P2PTV-AS automatically recovers the affected downloaders.

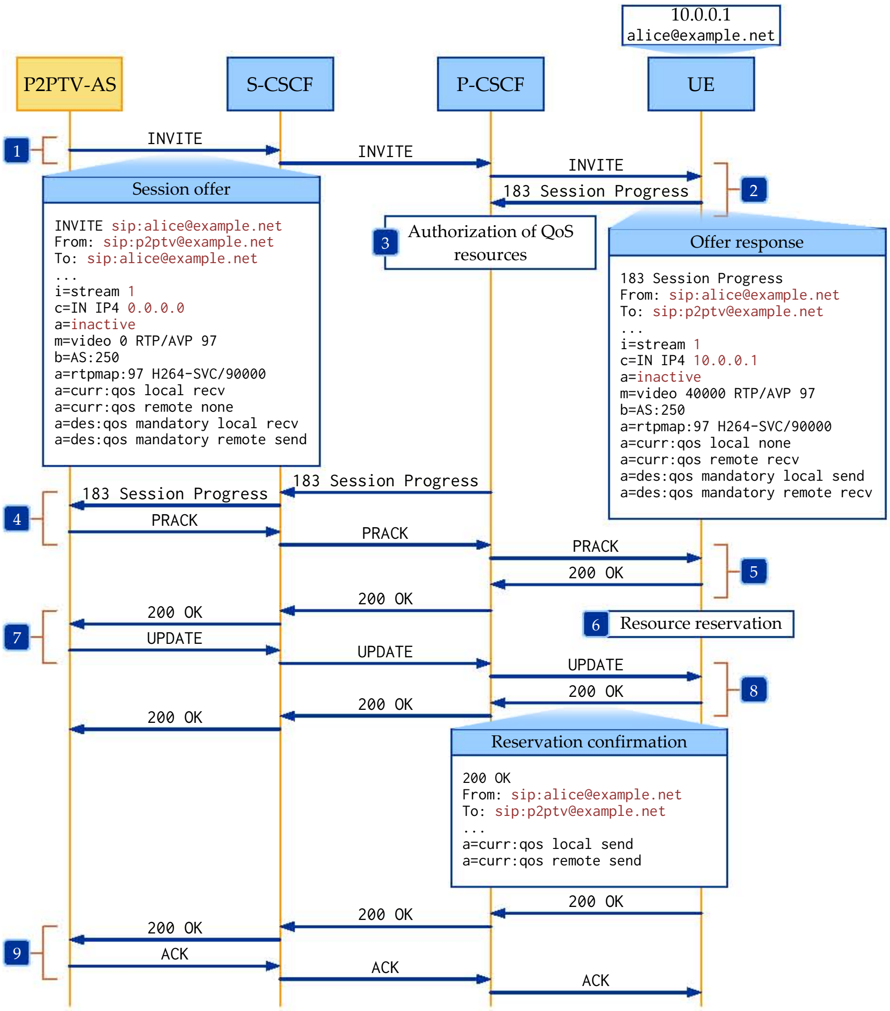
A.3.1 Upload Session Establishment

The P2PTV-AS establishes inactive upload sessions upon events generated by the SCA and the PCA (the SCA computes the required number of inactive sessions, while the PCA determines the UE peers that can service them). We do not regard active sessions as initiated by the P2PTV-AS, because they correspond to a request coming from a user equipment, as described in the procedures from A.2.1. The figure A.10 shows the steps involved.

1. Upon determining the necessity of a new inactive upload session allocated to the Alice's UE, the P2PTV-AS sends an `INVITE` request with the request URI and `To` SIP header set to the public user identity of Alice (`sip:alice@example.net`), and the `From` header set to public service identity of the P2PTV service (`sip:p2ptv@example.net`). The request contains an SDP body with the session information. Because the upload session is inactive, the SDP includes the `a=inactive` session attribute, whereas the connection (`c`) and media (`m`) fields may be set to unknown IP address and transport port values. However, the SDP body must contain the QoS precondition attributes, where the current QoS on the downloading party is assumed as established. As in the step 3 from the procedure A.2.1, the P2PTV-AS sends the request by including a list of `Route` headers with the URI of the S-CSCF assigned to Alice's UE, if the P2PTV-AS has routing capabilities. Otherwise, it may forward the request to its default S-CSCF[4].
2. Upon receiving the `INVITE` request, the Alice's UE verifies that the uploading conditions are met (the requested stream is available, the network conditions allow the upload, etc.), and replies with a `183 Session Progress` response containing the SDP response to the initial offer.
3. The P-CSCF assigned to Alice's UE uses the SDP information from the `183 Session Progress` reply to authorize the required QoS resources. The figure does not illustrate the signaling steps corresponding to this authorization, which conform to the procedure explained in the TS 29.214 (3GPP, 2012h). The P-CSCF forwards the reply message to P2PTV-AS via the S-CSCF.
4. The P2PTV-AS confirms the response, by sending a provisional acknowledge or `PRACK` request to Alice's UE.
5. The Alice's UE acknowledges the confirmation with a `200 OK` response.
6. Upon sending the `200 OK` response, the Alice's UE initiates the resource reservation procedure, according to its current access network.

[3] Generally, the design of the P2PTV-AS algorithms does not allow downloading peers to be disconnected from their uploaders, unless they completely turn off leaving the network (there is a single exception, when the service configuration allows peers to allocate bandwidth to secondary streams beyond the safe limit for primary channels). Since the probability of this occurrence is small, updating the download sessions is a small fraction of the session management operations initiated by the P2PTV-AS.

[4] Like before, we assume that the P2PTV-AS is a trusted part of the IMS core network, and the S-CSCFs recognize all requests initiated by the PSI of the P2PTV service.



Session offer

```
INVITE sip:alice@example.net
From: sip:p2ptv@example.net
To: sip:alice@example.net
...
i=stream 1
c=IN IP4 0.0.0.0
a=inactive
m=video 0 RTP/AVP 97
b=AS:250
a=rtpmap:97 H264-SVC/90000
a=curr:qos local recv
a=curr:qos remote none
a=des:qos mandatory local recv
a=des:qos mandatory remote send
```

Offer response

```
183 Session Progress
From: sip:alice@example.net
To: sip:p2ptv@example.net
...
i=stream 1
c=IN IP4 10.0.0.1
a=inactive
m=video 40000 RTP/AVP 97
b=AS:250
a=rtpmap:97 H264-SVC/90000
a=curr:qos local none
a=curr:qos remote recv
a=des:qos mandatory local send
a=des:qos mandatory remote recv
```

Reservation confirmation

```
200 OK
From: sip:alice@example.net
To: sip:p2ptv@example.net
...
a=curr:qos local send
a=curr:qos remote send
```

Figure A.10: The establishment of an inactive upload session.

7. When the P2PTV-AS received the provisional acknowledge confirmation, it sends a reservation confirmation UPDATE request to Alice's UE.

8. When the Alice's UE completes the resource reservation, and upon receiving the UPDATE request, it replies with a 200 OK response. The response contains an SDP body describing the updated QoS reservation.

9. Finally, the P2PTV-AS completes the session establishment procedure by acknowledging the reply with an ACK sent to Alice's UE.

Following the completion of the procedure, the P2PTV-AS updates the internal information of the session and peer coordination algorithms, to account for the newly established inactive session, where the update of the downloading sessions is described in the procedure A.3.4.

A.3.2 Upload Session Update

The P2PTV-AS updates an upload session in the following situations:

- during the UE-initiated update of a download session, according to the procedures from the section A.2.2;
- during the UE-initiated release of a download session, according to the procedure A.2.3, and;

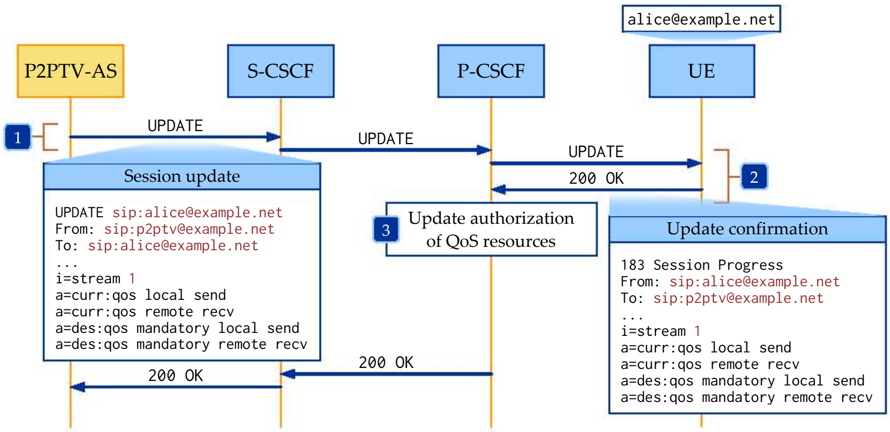

alice@example.net
P2PTV-AS
S-CSCF
P-CSCF
UE
1
UPDATE
UPDATE
UPDATE
200 OK
2
Session update
UPDATE sip:alice@example.net
From: sip:p2ptv@example.net
To: sip:alice@example.net
...
i=stream 1
a=curr:qos local send
a=curr:qos remote recv
a=des:qos mandatory local send
a=des:qos mandatory remote recv
3
Update authorization of QoS resources
Update confirmation
183 Session Progress
From: sip:alice@example.net
To: sip:p2ptv@example.net
...
i=stream 1
a=curr:qos local send
a=curr:qos remote recv
a=des:qos mandatory local send
a=des:qos mandatory remote recv
200 OK
200 OK

Figure A.11: The update of an upload session.

- during the recovery of an orphaned UE with the inactive session of a foster peer.

Usually, in the first two cases, the uploading session is changed from active to inactive and the uploading UE becomes a foster peer. In the latter situation, the session is changed from inactive to active. The figure A.11 illustrates the steps involved, which are the following.

1. The P2PTV-AS sends an `UPDATE` request, with the request URI and the `To` header set to the public user identity of Alice. The message contains an SDP body, including the updated session information such as the attribute (`a`), connection (`c`) and media (`m`) fields.

2. If the update is successful, the UE replies with a `200 OK` response.

3. Upon, receiving the update confirmation, the P-CSCF assigned to Alice's UE, updates the authorization of QoS resources.

Depending on the result of the update procedure, the Alice's UE may start or stop uploading the stream associated to the session.

A.3.3 Upload Session Release

The P2PTV-AS releases an upload session when the current number of foster peers is greater than the value computed by the session coordination algorithm. Under these circumstances, the PCA selects the peers that can have their upload session releases, according to the procedure from the figure A.12.

The steps are the following:

1. The P2PTV-AS sends a `BYE` request, with the request URI and the `To` header set to the public user identity of Alice.

2. Upon receiving the request, the P-CSCF assigned to Alice's UE removes the authorization for QoS resources.

3. The UE confirms the session release with a `200 OK` response.

4. The UE releases the allocated network resources, according to the type of access network.

We note that, in general, uploading UE peers do not release active media sessions directly, for instance when the corresponding downloading UE is leaving the stream by turning off or changing the channel. Instead, an uploading session is first updated from active to inactive, according to

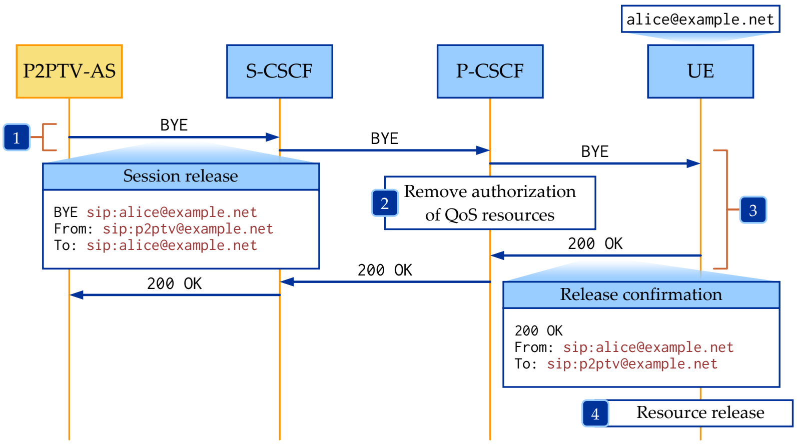

alice@example.net
P2PTV-AS
S-CSCF
P-CSCF
UE
1
BYE
BYE
BYE
Session release
BYE sip:alice@example.net
From: sip:p2ptv@example.net
To: sip:alice@example.net
2
Remove authorization of QoS resources
3
200 OK
200 OK
200 OK
Release confirmation
200 OK
From: sip:alice@example.net
To: sip:p2ptv@example.net
4
Resource release

Figure A.12: The release of an inactive upload session.

the procedure A.3.2 and synchronously with the update from the downloading peer, and then, if the uploading session is not required, is release according to the procedure from this section, synchronously with the clock of the SCA. This approach prohibits changing the number of upload session until the session coordination algorithm performs at least one computation cycle.

A.3.4 Download Session Update

The P2PTV-AS updates a download session during the recovery of an orphaned UE peer. The recovery may be both graceful and ungraceful. A graceful recovery occurs when the signaling procedures are executed according to the steps laid out in this appendix, and the stream to a downloading peer is never interrupted. An ungraceful recovery occurs during failures (software, network, electrical, etc.), which may be detected by both the P2PTV-AS and the downloading UE. The figure A.13 illustrates the steps involved in a download session update, which are the following.

1. The P2PTV-AS detects that it must update the current uploading entity assigned to Alice's UE, using one of the following mechanisms:

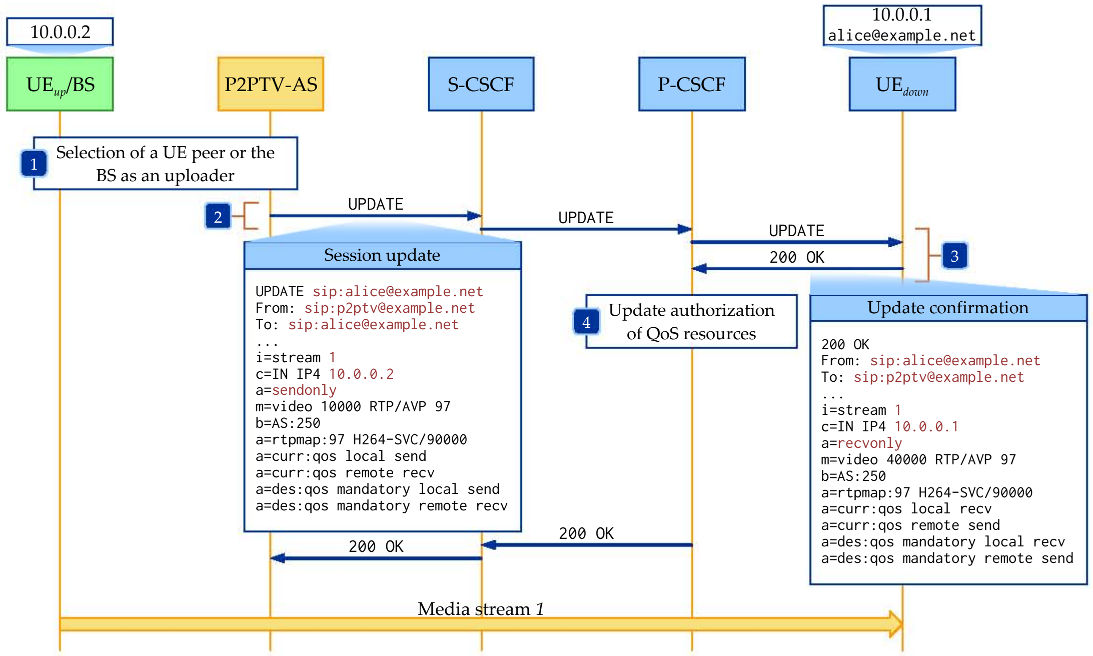
   - notification of session termination from the uploading UE and reported to the P2PTV-AS according to the procedure A.2.5.1;
   - notification of session failure from the downloading UE and reported to the P2PTV-AS according to the procedure A.2.5.2;
   - detection at the P2PTV-AS, via expired signaling timers or other mechanisms.

   Upon detection, the P2PTV-AS uses the PCA to assign a new uploading entity, a UE peer or the BS.

2. The P2PTV-AS sends an `UPDATE` to Alice's UE on the downloading session dialog that is being updated. The request includes an SDP body with the updated session information, including the connection (`c`) and the media (`m`) settings of the new uploading entity.

3. The UE updates the internal state, and replies to the P2PTV-AS with a `200 OK` response, including the updated session information.

4. Upon, receiving the update confirmation, the P-CSCF assigned to Alice's UE, updates the authorization of QoS resources.



```
UPDATE sip:alice@example.net
From: sip:p2ptv@example.net
To: sip:alice@example.net
...
i=stream 1
c=IN IP4 10.0.0.2
a=sendonly
m=video 10000 RTP/AVP 97
b=AS:250
a=rtpmap:97 H264-SVC/90000
a=curr:qos local send
a=curr:qos remote recv
a=des:qos mandatory local send
a=des:qos mandatory remote recv
```

```
200 OK
From: sip:alice@example.net
To: sip:p2ptv@example.net
...
i=stream 1
c=IN IP4 10.0.0.1
a=recvonly
m=video 40000 RTP/AVP 97
b=AS:250
a=rtpmap:97 H264-SVC/90000
a=curr:qos local recv
a=curr:qos remote send
a=des:qos mandatory local recv
a=des:qos mandatory remote send
```

Figure A.13: The update of a download session.

Following the completion of the update procedure, the UE begins receiving the media stream from the new uploading entity.

A.3.5 User Equipment Subscription

The P2PTV-AS uses a subscription dialog to receive notifications from the user equipments. We classify the UE notifications into:

- *action* notifications, which request the P2PTV-AS to initiate a procedure on behalf of the user equipment, such as a session termination (A.2.5.1) or session recovery (A.2.5.2), and;
- *information* notifications, which include the state parameters of the UE, such as total and available bandwidth (A.2.5.3).

The P2PTV-AS initiates the subscription dialog immediately after the completion of the UE subscription to the service information, described in the procedure A.2.4. The figure A.14 illustrates the steps involved, which are the following.

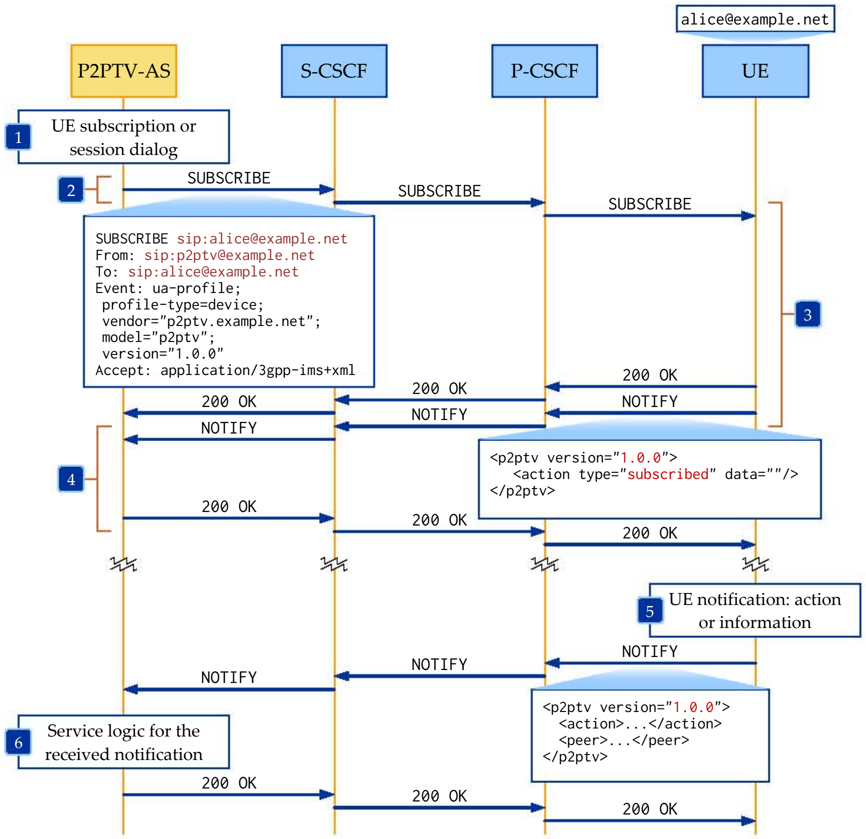
1. The UE completes the subscription to the P2PTV service information, according to the step 5 from the procedure A.2.4.

2. The P2PTV-AS sends a SUBSCRIBE request with the request URI set to the public user identity Alice. The subscription SIP message includes an Event header set to the ua-profile event package, with the following options:

   ```
   profile-type=device;vendor="p2ptv.example.net";model="p2ptv";version="1.0.0"
   ```

   The SUBSCRIBE request also includes an Accept header set to the value application/3gpp-ims+xml, indicating the MIME type bodies that are accepted in notifications.

3. Upon receiving the SUBSCRIBE request, the UE confirms the subscription by replying with a 200 OK response, and sending a NOTIFY request that includes an XML body, conforming to



```
SUBSCRIBE sip:alice@example.net
From: sip:p2ptv@example.net
To: sip:alice@example.net
Event: ua-profile;
 profile-type=device;
 vendor="p2ptv.example.net";
 model="p2ptv";
 version="1.0.0"
Accept: application/3gpp-ims+xml
```

```
<p2ptv version="1.0.0">
   <action type="subscribed" data=""/>
</p2ptv>
```

```
<p2ptv version="1.0.0">
  <action>...</action>
  <peer>...</peer>
</p2ptv>
```

Figure A.14: The AS subscription to the UE notifications.

the XML schema described in the section A.5, with at least one `action` element having the `type` attribute set to `subscribed`. Optionally, the UE may also include a UE peer information element, as specified in the section A.2.5.3.

4. The P2PTV-AS saves the UE subscription and peer information, if included, and uses it as input configuration for the peer coordination algorithm (PCA). The P2PTV-AS confirms the notification with a `200 OK` response.

5. At future instances, the UE uses the subscription dialog to inform the P2PTV-AS of certain events, as explained in the section A.2.5. To this end, the UE sends a `NOTIFY` request including an XML body with the notification information. Each notification may contain one or more `action` elements and/or a `peer` element.

6. The P2PTV-AS processes the notification message and, it replies with a `200 OK` response.

A.3.6 Service Information Notifications

The signaling procedure A.2.4 described the service information notification, when sent immediately after the establishment of the subscription dialog. In addition, the P2PTV-AS sends similar notifications to the user equipments, whenever the current configuration of the service, such as the available TV channels, changes. The software logic of the user equipments must be prepared to accept such notification, and update accordingly (i) the user interface including the list of TV channels displayed to the viewer, and (ii) the P2PTV layer regarding the TV channels that may be uploaded by the UE peer.

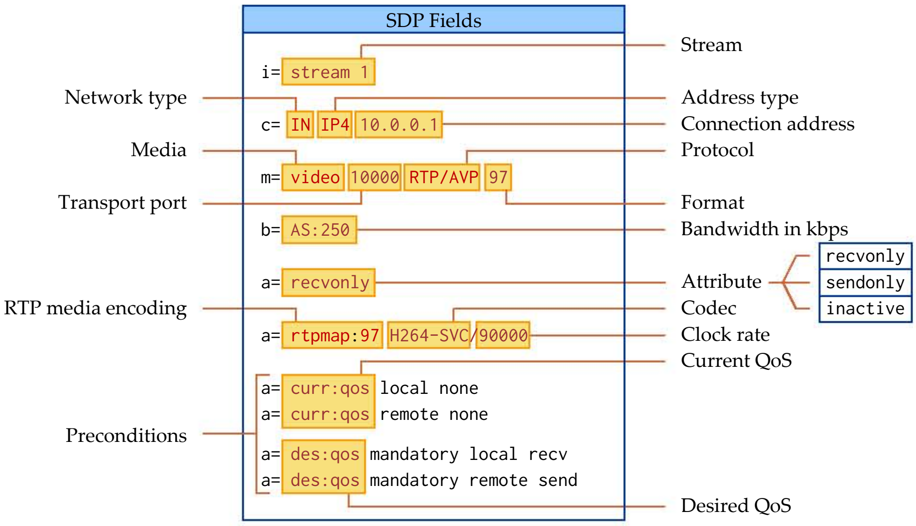


Figure A.15: The SDP fields required for the P2PTV session description.

A.4 Session Description

A.4.1 General Aspects

The IMS signaling uses the Session Description Protocol to exchange the session information between the uploading and downloading entities (Handley et al., 2006). In general, the SDP payload includes the IP address and transport ports, where the communication parties can receive data, such as RTP and RTCP packets, the media encoding format, and bandwidth information. All these parameters are used by both the user equipment, to allocate the necessary encoding/decoding capabilities to process the stream data, and the IMS core to perform the necessary authorization of QoS resources according to the mapping from TS 29.213 (3GPP, 2012i).

In addition to the mandatory fields required by the standard, we include the session information field (i), which describes the session stream. This parameter allows the decoder to match different streams belonging to the same TV channel, irrespective of the underlying video codec. We note that some existing codecs, such as the H.264/SVC, already provide a standardized alternative to including stream information. In the next section, we provide an example of the SDP structure, when using scalable video coding.

The figure A.15 and the table A.1 summarize the fields required in the SDP body during a session establishment or update. The specifications of the audio and video encodings using RTP/RTCP transport are further standardized in RFC 3551 (Schulzrinne & Casner, 2003), RFC 5761 (Perkins & Westerlund, 2010), RFC 6184 for H.624/AVC (Y. K. Wang, Kristensen, & Jesup, 2011) and RFC 6190 for H.624/SVC (Y. K. Wang, Schierl, & Eleftheriadis, 2011).

A.4.2 Session Description for H.264 Scalable Video Coding

In this work, we proposed the existing Appendix G of the H.264/AVC standard, usually referred to as Scalable Video Coding or SVC, as a possible implementation to the division of a TV channels into multiple streams. In practice, this division is necessary to accommodate the variable granularity of UE peer download and upload bandwidth. Depending on the type of scalability used, SVC streams share a dependency relationship, that allows the decoder to recover a higher-layer video data only after the lower-layer streams were received.

From the perspective of a general codec, the user equipments use the channels/streams service information, broadcasted by the P2PTV-AS over the subscription dialogs, to determine how

Field	Description	Observation
`i`	Stream	See RFC 4566, section 5.4 (Handley et al., 2006).
`c`	Connection data	See RFC 4566, section 5.7 (Handley et al., 2006).
`m`	Media description	See RFC 4566, section 5.14 (Handley et al., 2006).
`b`	Bandwidth (in kbps)	See RFC 3890 (Westerlund, 2004).
`a=recvonly`	Session is receive only	See RFC 4566, section 6 (Handley et al., 2006).
`a=sendonly`	Session is send only	See RFC 4566, section 6 (Handley et al., 2006).
`a=inactive`	Session is inactive	See RFC 4566, section 6 (Handley et al., 2006).
`a=rtpmap`	RTP media encoding	See RFC 4566, section 6 (Handley et al., 2006).

Table A.1: The SDP fields required for the P2PTV session description.

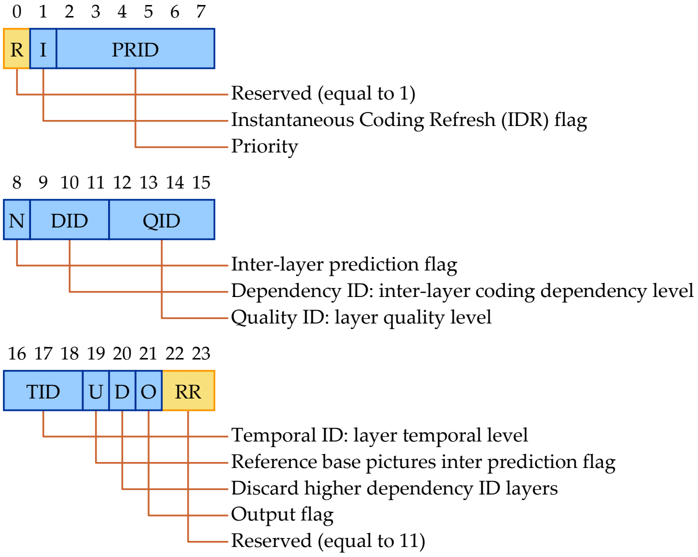

0 1 2 3 4 5 6 7
R
I
PRID
Reserved (equal to 1)
Instantaneous Coding Refresh (IDR) flag
Priority
8 9 10 11 12 13 14 15
N
DID
QID
Inter-layer prediction flag
Dependency ID: inter-layer coding dependency level
Quality ID: layer quality level
16 17 18 19 20 21 22 23
TID
U
D
O
RR
Temporal ID: layer temporal level
Reference base pictures inter prediction flag
Discard higher dependency ID layers
Output flag
Reserved (equal to 11)

Figure A.16: The Network Abstraction Layer (NAL) unit header extension for H.264 Scalable Video Coding.

different streams correspond to a TV channel. Then, during session establishment, the user equipments includes the stream information in the information field (`i`) of the SDP payload. In turn, we assume that there exists an appropriate mechanism in place, which allows the UE software codecs to make the correct mapping between a stream and the TV channel decoded data[5].

When using H.264/SVC (ITU-T, 2011), the RFC 6190 offers a more elegant alternative to including stream information directly into the SDP payload (Y. K. Wang, Schierl, & Eleftheriadis, 2011). These parameters, presented as session attribute fields (a), describe the properties of the SVC stream and its transmission using the RTP (Schulzrinne et al., 2003). Without entering into too much detail, the SVC standard maintains the stream organization, with video components encapsulated in Network Abstraction Layer, or NAL, units. Compared to the AVC standard, the SVC specification extends the original header to include the scalability information. This 3-byte extension, illustrated in the figure A.16, includes the stream layer level into each of the possible scalability dimensions: temporal, spatial and quality (or signal-to-noise ratio).

The same standard mandates several extensions to the SDP format, which include the descrip-

[5] Such a mechanism would include a built-in configuration, additional settings in the subscribed service information, or decoding instructions in the stream data.

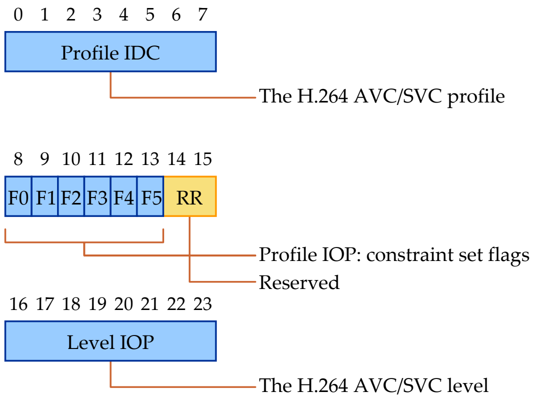


Figure A.17: The `profile-level-id` parameter describing an SVC stream session.

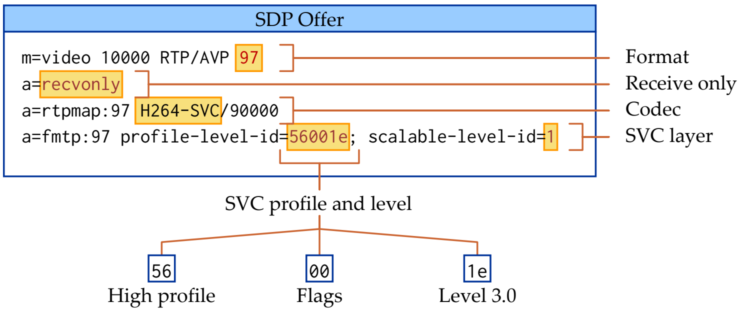


Figure A.18: The SDP offer for receiving a H.264/SVC stream.

tion of SVC streams. One of the new parameters is the `profile-level-id` with a 3-byte value in hexadecimal notation, conforming to the structure from the figure A.17, describing the SVC profile and level (ITU-T, 2011). The table A.2 summarizes the profiles allowed by the SVC standard, where generally broadcast television requires the *scalable high* profile.

SVC Profile	Value	Profile IOP Flags	Description
Scalable Baseline	83 (0x53)	x0000000	Video-conference/surveillance
Scalable High	86 (0x56)	0x000000	Video-broadcast
Scalable High Intra	86 (0x56)	0x010000	Professional applications

Table A.2: The profiles defined by the H.264/SVC standard.

In addition to the `profile-level-id`, in the P2PTV service, the user equipments use the `scalable-level-id` parameter to request a specific layer of the SVC coded video. In the figure A.18, we show an SDP offer example, including the minimum number of SDP fields.

A.4.3 Support for Advanced Peer-to-Peer Features

Unlike video streaming from a broadcast server, the peer-to-peer approach poses unique challenges. In particular, the peer churn is a significant performance-limiting factor. Unfortunately, the present specifications do not consider thoroughly the specific requirements of peer-to-peer streaming. For example, the streaming-boost method requires sharing the same stream between two different sources for a limited duration, in order to augment the buffered data at the receiving peer. It is not the purpose of our present work to propose new extensions to the SDP standards, which cover this advanced P2P features. However, we do note that future implementations should consider such extensions.

A.5 XML Schema for Service Notifications

The code samples A.4–A.5, and the figure A.19 describe the XML schema used for the XML body of service and UE notifications.

Code sample A.4: The XML schema for service notifications.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified" version="1">
  <xs:complexType name="tIMS3GPP">
    <xs:sequence>
      <xs:element name="p2ptv" type="tP2PTV"/>
    </xs:sequence>
    <xs:attribute name="version" type="xs:string" use="required"/>
    <xs:anyAttribute/>
  </xs:complexType>
  <xs:complexType name="tP2PTV">
    <xs:sequence>
      <xs:element name="action" type="tP2PTVAction" minOccurs="0"
                  maxOccurs="unbounded"/>
      <xs:element name="service" type="tP2PTVService"/>
      <xs:element name="peer" type="tP2PTVPeer"/>
    </xs:sequence>
    <xs:attribute name="version" type="xs:decimal"/>
  </xs:complexType>
  <xs:complexType name="tP2PTVAction">
    <xs:sequence>
      <xs:element name="stream" type="xs:string" minOccurs="1"
                  maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="type" type="xs:string"/>
    <xs:attribute name="data" type="xs:string"/>
  </xs:complexType>
  <xs:complexType name="tP2PTVService">
    <xs:sequence>
      <xs:element name="channels" type="tP2PTVChannels"/>
      <xs:element name="streams" type="tP2PTVStreams"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="tP2PTVPeer">
    <xs:sequence>
      <xs:element name="bwtotal" type="tP2PTVBandwidth"/>
      <xs:element name="bwfree" type="tP2PTVBandwidth"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="tP2PTVChannels">
    <xs:sequence>
      <xs:element name="channel" type="tP2PTVChannel" minOccurs="0"
                  maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="tP2PTVStreams">
    <xs:sequence>
      <xs:element name="stream" type="tP2PTVStream" minOccurs="0"
                  maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="tP2PTVChannel">
    <xs:sequence>
      <xs:element name="stream" type="xs:decimal" minOccurs="1"
                  maxOccurs="unbounded"/>
```

Code sample A.5: The XML schema for service notifications (continued).

```
    </xs:sequence>
    <xs:attribute name="id" type="xs:decimal"/>
    <xs:attribute name="name" type="xs:string"/>
  </xs:complexType>
  <xs:complexType name="tP2PTVStream">
    <xs:sequence>
      <xs:element name="bandwidth" type="xs:decimal"/>
      <xs:element name="audio" type="tP2PTVStreamData"/>
      <xs:element name="video" type="tP2PTVStreamData"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:decimal"/>
    <xs:attribute name="channel" type="xs:decimal"/>
  </xs:complexType>
  <xs:complexType name="tP2PTVStreamData">
    <xs:sequence>
      <xs:element name="param" type="tP2PTVParam" minOccurs="0"
                  maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="codec" type="xs:string"/>
  </xs:complexType>
  <xs:complexType name="tP2PTVParam">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="name" type="xs:string"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="tP2PTVBandwidth">
    <xs:sequence>
      <xs:element name="upload" type="xs:decimal"/>
      <xs:element name="download" type="xs:decimal"/>
    </xs:sequence>
  </xs:complexType>
  <!-- Root Element -->
  <xs:element name="ims-3gpp" type="tIMS3GPP"/>
  <xs:element name="type" type="xs:string"/>
</xs:schema>
```

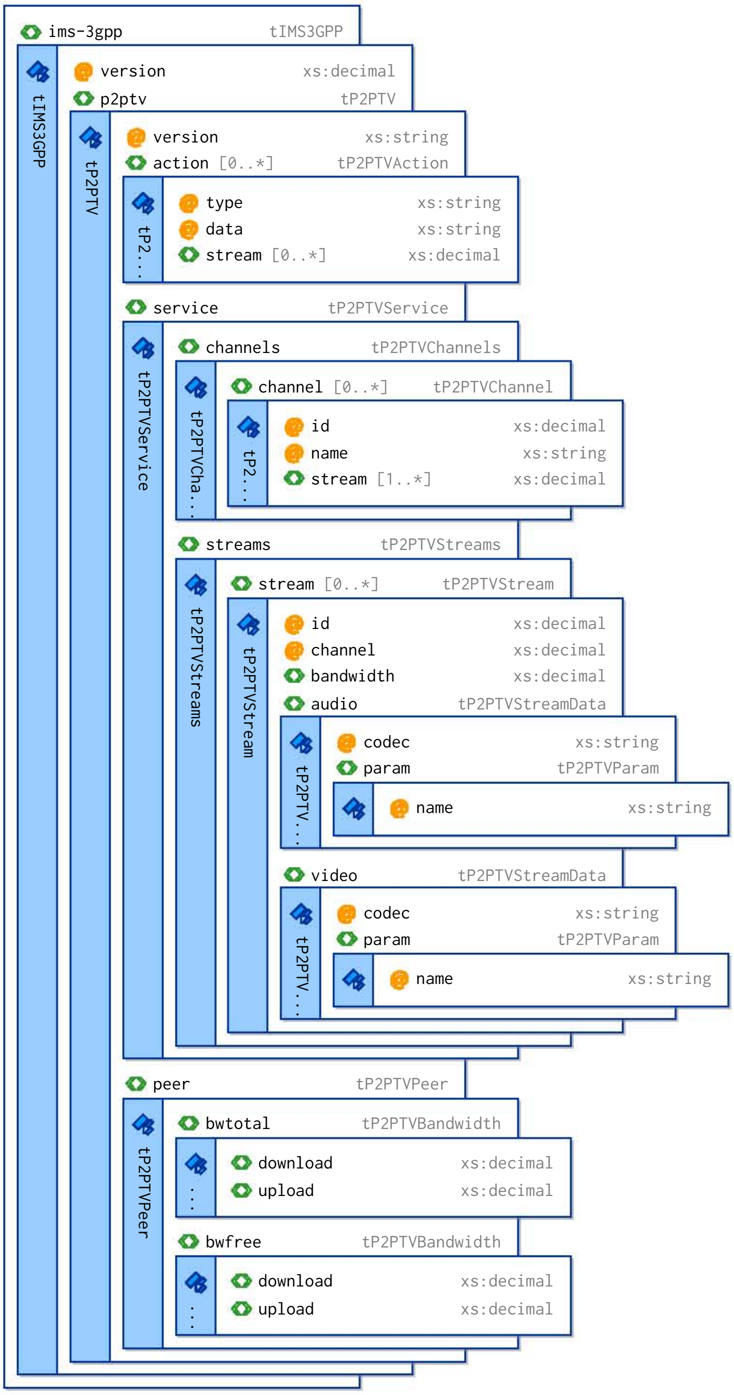


Figure A.19: The XML schema for service notifications.

Appendix B

Service Profile Configuration

B.1 Overview

In order to access the P2PTV service, a user requires an appropriate service profile in his or her IMS subscription. In essence, this represents a mapping between the user's private and public identities, configured at the user equipment (UE), and the initial filter criteria (iFC) corresponding to the P2PTV service. The user and service profiles are stored in the HSS/UPSF in the form of a relational database. In general, an IMS user may have several private and public identities, as we show in the figure B.1, where a service profile can be attached to one or more of these identities.

During the initial user registration, usually performed automatically when the user turns on the set-top box UE, the service profile corresponding to the current identity is accessed by the S-CSCF. The S-CSCF uses this data to provide a specific service during each multimedia session, by verifying the SIP request messages against the list of triggers from the iFC. When the trigger conditions are met, the SIP messages are routed to the configured application server. The figure B.2 shows the simplified structure of a service profile, containing multiple filter criteria (3GPP, 2012b).

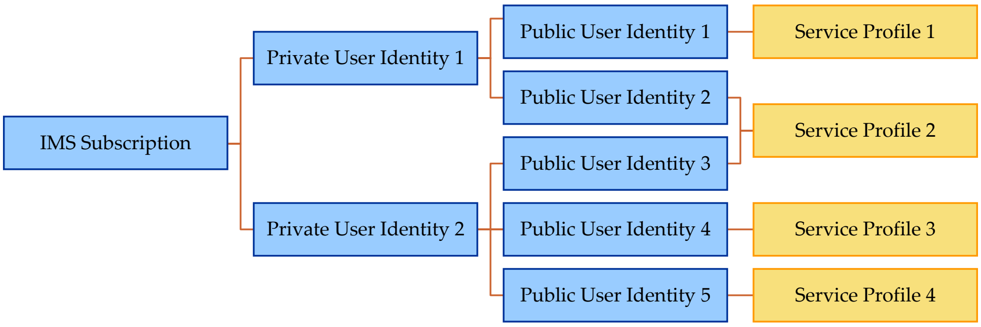
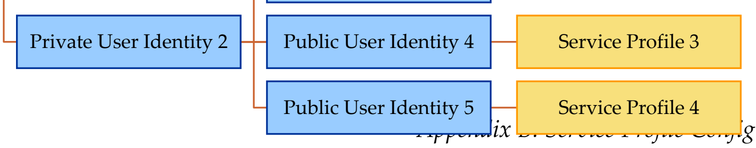


Figure B.1: The general form of an IMS subscription.



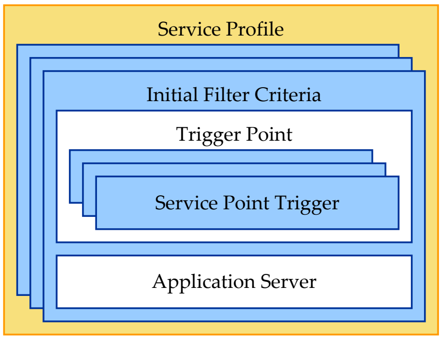


Figure B.2: The general form of an IMS service profile.

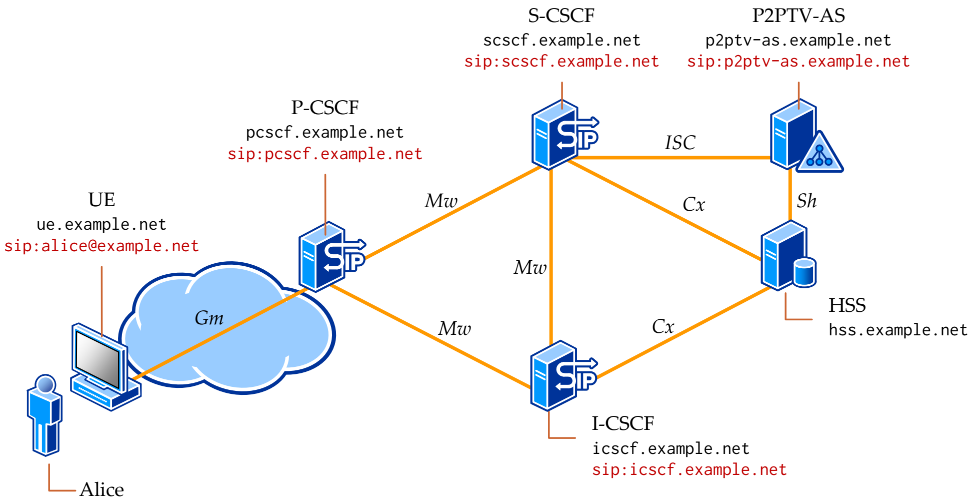


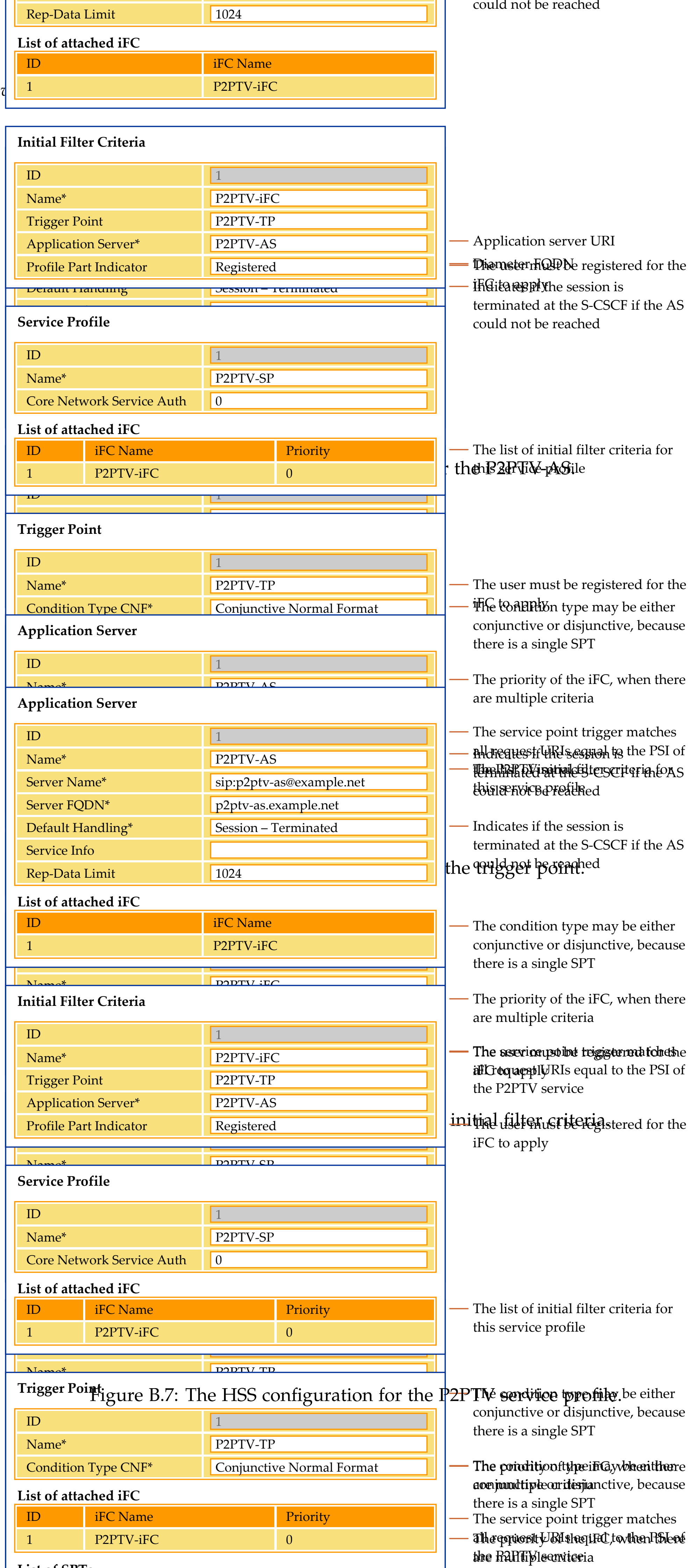
Figure B.3: The schematic of the service profile testbed.

In this appendix, we detail the specific configuration of the P2PTV service, which is required for both the service profile and user subscriptions. For the reasons of simplicity, we shall assume that our users have a single private and public identity, and are only subscribed to the P2PTV service. However, we believe that extending our toy configuration is relatively straightforward.

The figure B.3 illustrates the schematic of our testbed in a realm with the DNS suffix `example.net`, with a P2PTV subscriber, named *Alice*, whose private and public users identities are `alice@example.net` and `sip:alice@example.net`, respectively. The public service identity (PSI) of the P2PTV service is `sip:p2ptv@example.net`. In addition, the figure shows the fully qualified domain names (FQDN) and SIP URIs of the main IMS core network functions: the P-CSCF, the I-CSCF, the S-CSCF, the HSS and the P2PTV-AS. The IMS core network uses the FOKUS OpenIMSCore open-source implementation of the IMS functional entities[1].

The setup of the service profile and user subscription uses the web interface of the HSS. For these reasons, our example focuses on the configuration options available in the FOKUS software and although the most important parameters should be the same, we acknowledge that there may be differences to other HSS implementations.

[1] FOKUS OpenIMSCore, http://www.openimscore.org/ .



Rep-Data Limit	1024

List of attached iFC

ID	iFC Name
1	P2PTV-iFC

could not be reached

Initial Filter Criteria

ID	1
Name*	P2PTV-iFC
Trigger Point	P2PTV-TP
Application Server*	P2PTV-AS
Profile Part Indicator	Registered

Application server URI

Diameter FQDN

The user must be registered for the iFC to apply

Indicates if the session is terminated at the S-CSCF if the AS could not be reached

Service Profile

ID	1
Name*	P2PTV-SP
Core Network Service Auth	0

List of attached iFC

ID	iFC Name	Priority
1	P2PTV-iFC	0

The list of initial filter criteria for this service profile

the P2PTV AS.

Trigger Point

ID	1
Name*	P2PTV-TP
Condition Type CNF*	Conjunctive Normal Format

The user must be registered for the iFC to apply

The condition type may be either conjunctive or disjunctive, because there is a single SPT

The priority of the iFC, when there are multiple criteria

Application Server

ID	1
Name*	P2PTV-AS

The service point trigger matches all request URIs equal to the PSI of the P2PTV service

Indicates if the session is terminated at the S-CSCF if the AS could not be reached

The list of initial filter criteria for this service profile

Application Server

ID	1
Name*	P2PTV-AS
Server Name*	sip:p2ptv-as@example.net
Server FQDN*	p2ptv-as.example.net
Default Handling*	Session – Terminated
Service Info	
Rep-Data Limit	1024

List of attached iFC

ID	iFC Name
1	P2PTV-iFC

Indicates if the session is terminated at the S-CSCF if the AS could not be reached

the trigger point.

The condition type may be either conjunctive or disjunctive, because there is a single SPT

The priority of the iFC, when there are multiple criteria

Initial Filter Criteria

ID	1
Name*	P2PTV-iFC
Trigger Point	P2PTV-TP
Application Server*	P2PTV-AS
Profile Part Indicator	Registered

The service point trigger matches all request URIs equal to the PSI of the P2PTV service

initial filter criteria.

The user must be registered for the iFC to apply

Service Profile

ID	1
Name*	P2PTV-SP
Core Network Service Auth	0

List of attached iFC

ID	iFC Name	Priority
1	P2PTV-iFC	0

The list of initial filter criteria for this service profile

Trigger Point

ID	1
Name*	P2PTV-TP
Condition Type CNF*	Conjunctive Normal Format

List of attached iFC

ID	iFC Name	Priority
1	P2PTV-iFC	0

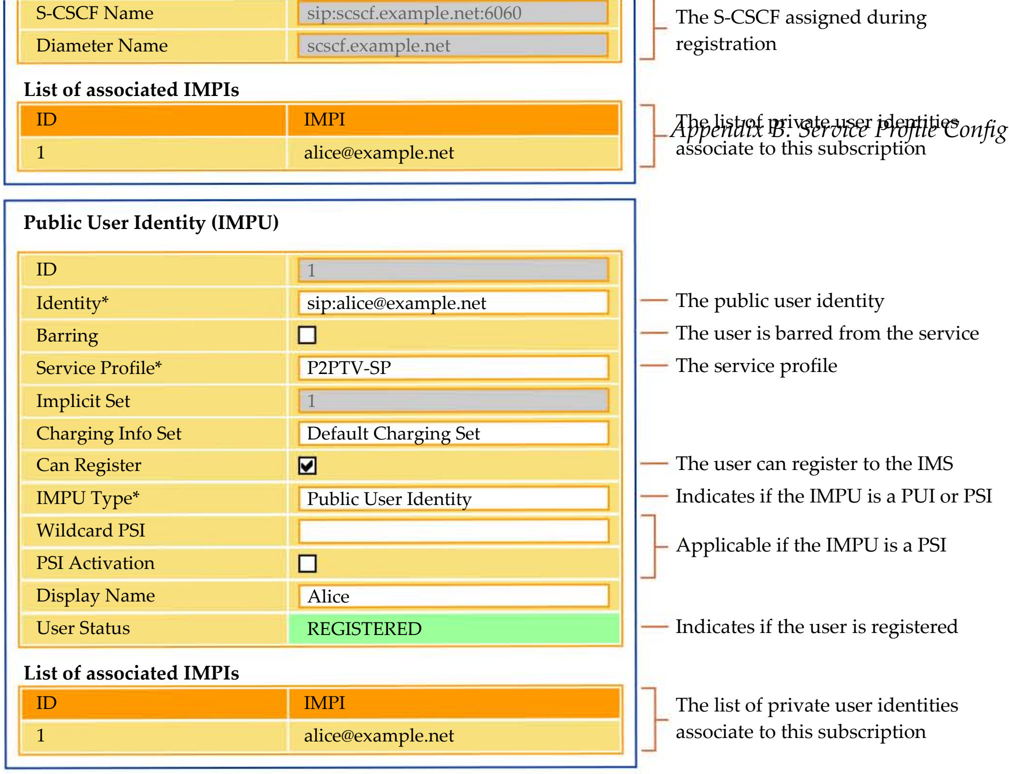
List of SPTs

The condition type may be either conjunctive or disjunctive, because there is a single SPT

The priority of the iFC, when there are multiple criteria

The service point trigger matches all request URIs equal to the PSI of the P2PTV service

Figure B.7: The HSS configuration for the P2PTV service profile.



S-CSCF Name	sip:scscf.example.net:6060
Diameter Name	scscf.example.net

The S-CSCF assigned during registration

List of associated IMPIs

ID	IMPI
1	alice@example.net

The list of private user identities associate to this subscription

Public User Identity (IMPU)

ID	1
Identity*	sip:alice@example.net
Barring	□
Service Profile*	P2PTV-SP
Implicit Set	1
Charging Info Set	Default Charging Set
Can Register	☑
IMPU Type*	Public User Identity
Wildcard PSI	
PSI Activation	□
Display Name	Alice
User Status	REGISTERED

The public user identity

The user is barred from the service

The service profile

The user can register to the IMS

Indicates if the IMPU is a PUI or PSI

Applicable if the IMPU is a PSI

Indicates if the user is registered

List of associated IMPIs

ID	IMPI
1	alice@example.net

The list of private user identities associate to this subscription

Figure B.8: The public user identity (IMPU) of Alice.

B.2 Service Profile

The service profile includes the definition of the application server, the trigger point and the initial filter criteria. The figure B.4 illustrates the parameters of the P2PTV-AS. Besides the expected values, including a symbolic name, the SIP URI and the Diameter FQDN (required for the *Sh* reference point between the AS and the HSS), the default handling option indicates whether the S-CSCF should terminate or continue processing a SIP dialog when the AS is unreachable.

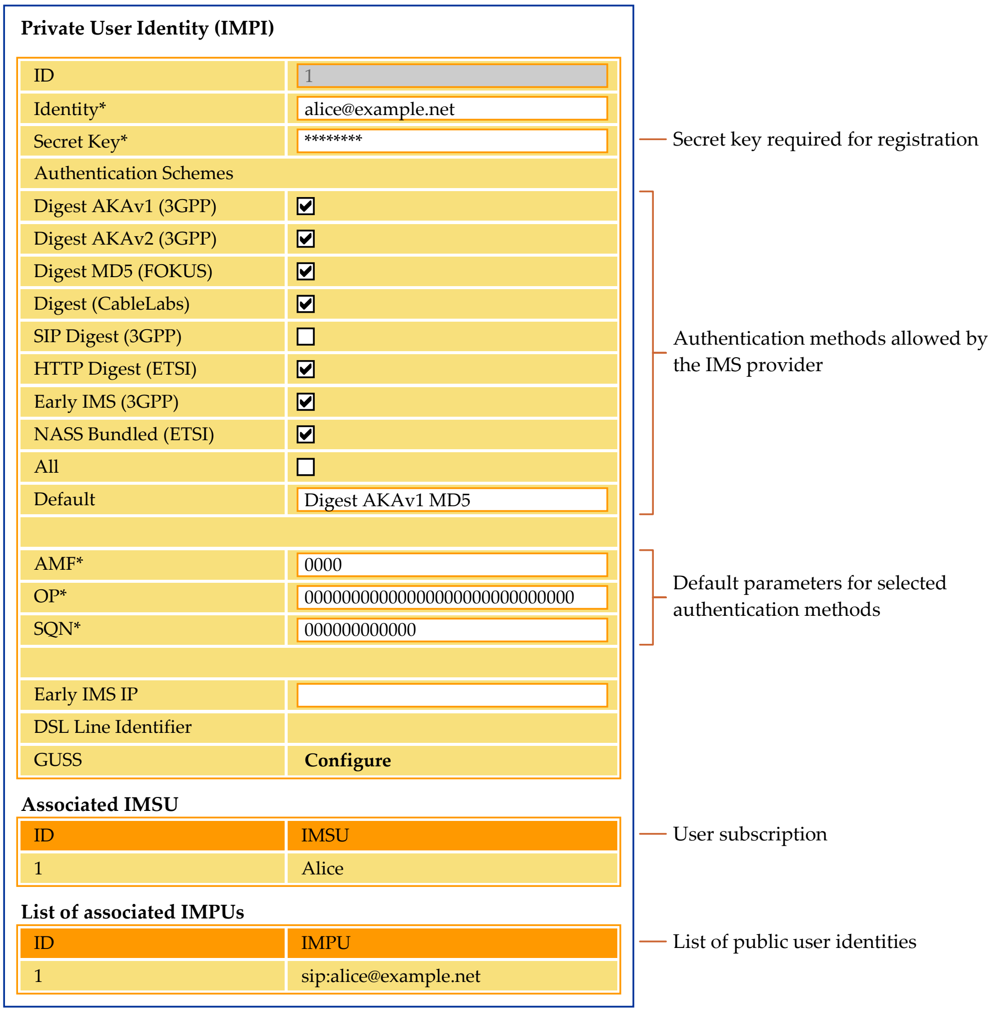
The filter criteria trigger point, depicted in the figure B.5, contains a list of conditions called service point triggers, or SPTs. These conditions can be represented in a conjunctive or disjunctive format, where in conjunctive form the expressions are first or-ed and then and-ed, and the disjunctive, the other way around. We propose using a single SPT that matches the request URI of a SIP message to the PSI of the P2PTV service, and therefore both formats are equivalent.

Finally, the filter criteria, shown in the figure B.6, represents a mapping between the list of service point triggers and the application server, whereas the service profile contains an ordered list of filter criteria. In our example, from the figure B.7, we have only the P2PTV service iFC with priority number zero. Multiple filter criteria are executed in the order of their priority number, where a lower number having a higher priority.

B.3 User Subscription

The user subscription (IMSU) includes the private and public user identities, abbreviated as IMPI and IMPU, respectively. First, we define the URI string of the public user *identity*, set to `sip:alice@example.net`, and link it to the service profile including the P2PTV service, as shown in the figure B.8. When accessing the P2PTV service, the user Alice must use the identity that is mapped to the profile including the P2PTV service. The boolean *barring* property is set to false, and it specified whether the S-CSCF allows the public identity to be used for establishing multimedia sessions. When the barring property is true, the public user identity may only be used for registration and re-registration purposes. Similarly, the *can register* is set to true, and indicates that the same IMPU can be used for registration.

Second, we define the private user *identity* (IMPI) that includes the *secret key* and the allowed



Private User Identity (IMPI)

ID	1
Identity*	alice@example.net
Secret Key*	********
Authentication Schemes	
Digest AKAv1 (3GPP)	☑
Digest AKAv2 (3GPP)	☑
Digest MD5 (FOKUS)	☑
Digest (CableLabs)	☑
SIP Digest (3GPP)	□
HTTP Digest (ETSI)	☑
Early IMS (3GPP)	☑
NASS Bundled (ETSI)	☑
All	□
Default	Digest AKAv1 MD5
AMF*	0000
OP*	00000000000000000000000000000000
SQN*	000000000000
Early IMS IP	
DSL Line Identifier	
GUSS	**Configure**

Associated IMSU

ID	IMSU
1	Alice

List of associated IMPUs

ID	IMPU
1	sip:alice@example.net

Secret key required for registration

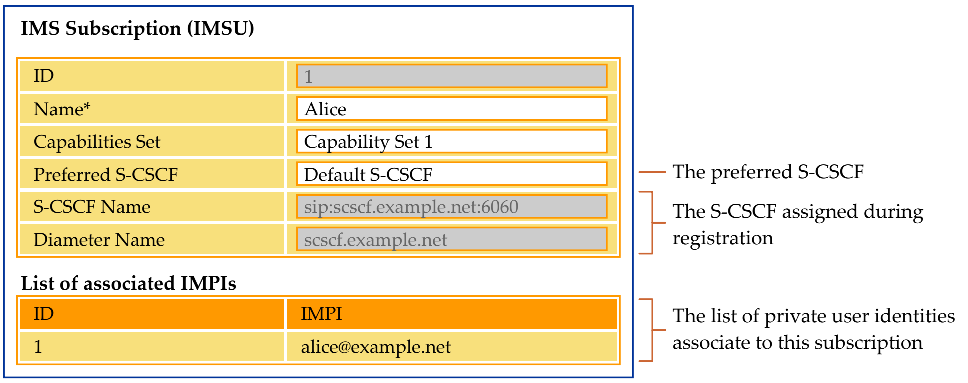
Authentication methods allowed by the IMS provider

Default parameters for selected authentication methods

User subscription

List of public user identities

Figure B.9: The private user identity (IMPI) of Alice.



IMS Subscription (IMSU)

ID	1
Name*	Alice
Capabilities Set	Capability Set 1
Preferred S-CSCF	Default S-CSCF
S-CSCF Name	sip:scscf.example.net:6060
Diameter Name	scscf.example.net

List of associated IMPIs

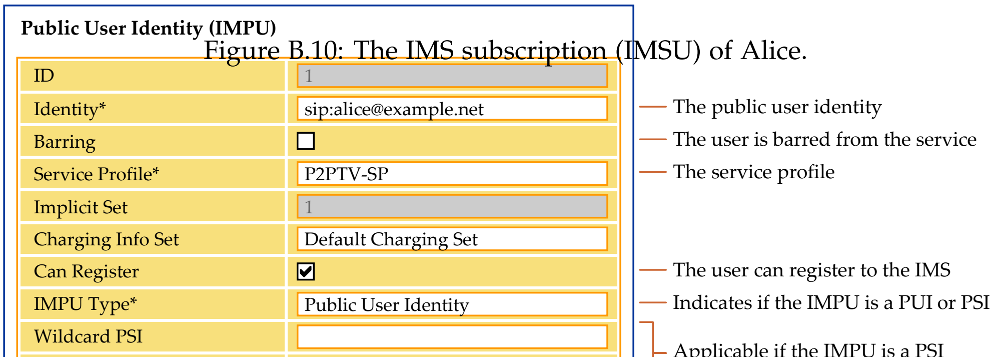
ID	IMPI
1	alice@example.net

The preferred S-CSCF

The S-CSCF assigned during registration

The list of private user identities associate to this subscription

Figure B.10: The IMS subscription (IMSU) of Alice.



Public User Identity (IMPU)

ID	1
Identity*	sip:alice@example.net
Barring	□
Service Profile*	P2PTV-SP
Implicit Set	1
Charging Info Set	Default Charging Set
Can Register	☑
IMPU Type*	Public User Identity
Wildcard PSI	

The public user identity

The user is barred from the service

The service profile

The user can register to the IMS

Indicates if the IMPU is a PUI or PSI

Applicable if the IMPU is a PSI

authentication schemes for the IMS registration. In our example, from the figure B.9, we leave these options set to the default values. In practice, these settings may consider the actual capabilities of the P2PTV clients and the requirements of the IMS provider. The private identity is associated to the public identity, such that it can be used to authenticate the user. Finally, we define the IMS user subscription in the figure B.10, which includes the public and the private identities, the preferred S-CSCF and the list of S-CSCF capabilities. The FOKUS HSS web interface informs the operator of the registration state of the user, as well as the assigned S-CSCF.

Appendix C

Design Elements for the Session Coordination Algorithm

C.1 Closed-Loop Poles for the Fast Control Algorithm

Given the PI controller characterized by the feedback gain κ and the open-loop zero z_0, the poles of the closed-loop system are:

$$\begin{aligned} p_{c,1} &= -\frac{1}{2}(\kappa - 1) + \frac{1}{2}\sqrt{(\kappa - 1)^2 + 4\kappa z_0}, \\ p_{c,2} &= -\frac{1}{2}(\kappa - 1) - \frac{1}{2}\sqrt{(\kappa - 1)^2 + 4\kappa z_0}. \end{aligned} \tag{C.1}$$

For a positive zero $0 \leq z_0 \leq 1$, the limit of the closed-loop poles when the gain goes from zero to infinity are the following. For the first closed-loop pole $p_{c,1}$, we have:

$$\begin{aligned} \lim_{\kappa \to 0} p_{c,1} &= \lim_{\kappa \to 0} \left(-\frac{1}{2}(\kappa - 1) + \frac{1}{2}\sqrt{(\kappa - 1)^2 + 4\kappa z_0} \right) \\ &= \frac{1}{2} + \frac{1}{2} \\ &= 1, \end{aligned} \tag{C.2}$$

and:

$$\begin{aligned}
\lim_{\kappa\to\infty} p_{c,1} &= \lim_{\kappa\to\infty}\left(-\frac{1}{2}(\kappa-1)+\frac{1}{2}\sqrt{(\kappa-1)^2+4\kappa z_0}\right)\\
&= \frac{1}{2}\lim_{\kappa\to\infty}\left(\sqrt{(\kappa-1)^2+4\kappa z_0}-(\kappa-1)\right)\\
&= \frac{1}{2}\lim_{\kappa\to\infty}\left[\left(\sqrt{(\kappa-1)^2+4\kappa z_0}-(\kappa-1)\right)\frac{\sqrt{(\kappa-1)^2+4\kappa z_0}+(\kappa-1)}{\sqrt{(\kappa-1)^2+4\kappa z_0}+(\kappa-1)}\right]\\
&= \frac{1}{2}\lim_{\kappa\to\infty}\frac{(\kappa-1)^2+4\kappa z_0-(\kappa-1)^2}{\sqrt{(\kappa-1)^2+4\kappa z_0}+(\kappa-1)}\\
&= \frac{1}{2}\lim_{\kappa\to\infty}\frac{4\kappa z_0}{\sqrt{(\kappa-1)^2+4\kappa z_0}+(\kappa-1)}.
\end{aligned} \tag{C.3}$$

Since the limits of both the numerator and denominator of the previous fraction are infinite, we can apply the l'Hôpital's rule, and we have:

$$\begin{aligned}
\lim_{\kappa\to\infty} p_{c,1} &= \frac{1}{2}\lim_{\kappa\to\infty}\frac{4z_0}{\frac{(\kappa-1)+4z_0}{\sqrt{(\kappa-1)^2+4\kappa z_0}}+1}\\
&= \frac{4z_0}{2\left(\lim_{\kappa\to\infty}\sqrt{\frac{(\kappa-1)^2+4(\kappa-1)z_0+16z_0^2}{(\kappa-1)^2+4\kappa z_0}}+1\right)}\\
&= \frac{4z_0}{2(1+1)}\\
&= z_0.
\end{aligned} \tag{C.4}$$

For the second closed-loop pole $p_{c,2}$, we have:

$$\begin{aligned}
\lim_{\kappa\to 0} p_{c,2} &= \lim_{\kappa\to 0}\left(-\frac{1}{2}(\kappa-1)-\frac{1}{2}\sqrt{(\kappa-1)^2+4\kappa z_0}\right)\\
&= \frac{1}{2}-\frac{1}{2}\\
&= 0,
\end{aligned} \tag{C.5}$$

and:

$$\begin{aligned}
\lim_{\kappa\to\infty} p_{c,2} &= \lim_{\kappa\to 0}\left(-\frac{1}{2}(\kappa-1)-\frac{1}{2}\sqrt{(\kappa-1)^2+4\kappa z_0}\right)\\
&= -\infty-\infty\\
&= -\infty.
\end{aligned} \tag{C.6}$$

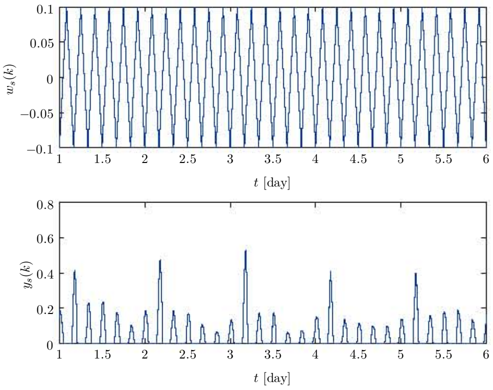


Figure C.1: Identification of the IPTV system response to the slow control algorithm.

C.2 System Identification for the Slow Control Algorithm

Let us assume that the P2PTV system response to the slow control algorithm is a first order system, described by the difference equation:

$$y_s(k+1) = a_s y_s(k) + b_s w_s(k). \tag{C.7}$$

The parameters a_s and b_s can be determined experimentally using the *least squares regression* method. The idea of the method is the following. Let $\tilde{w}_s$ and $\tilde{y}_s$ be the measured input and output, respectively. For an accurate description of the system around the operating point, the input is generally tuned manually. Our objective is to minimize the sum of squared errors around a desired operating point $(\bar{w}_s, \bar{y}_s)$, considering the offset input $w_s(k) = \tilde{w}_s(k) - \bar{w}_s$ and the offset output $y_s(k) = \tilde{y}_s(k) - \bar{y}_s$. The figure C.1 illustrates the measured output, when the input is a triangular signal of amplitude 0.1 around a null operating point.

Given the estimated values of a_s and b_s, the predicted output at moment $k+1$ is:

$$\hat{y}_s(k+1) = a_s y_s(k) + b_s w_s(k), \tag{C.8}$$

with an error:

$$e(k+1) = y_s(k+1) - \hat{y}_s(k+1) = y_s(k+1) - a_s y_s(k) - b_s w_s(k). \tag{C.9}$$

For a set of $N+1$ observations, the sum of squared errors is:

$$J(a_s, b_s) = \sum_{i=1}^{N} e^2(k+1) = \sum_{i=1}^{N} \left(y_s(k+1) - a_s y_s(k) - b_s w_s(k)\right)^2. \tag{C.10}$$

To find the values (a_s, b_s) where $J(a_s, b_s)$ has a minimum, we determine the roots of the partial derivatives equal to zero:

$$\begin{aligned} \frac{\partial}{\partial a_s} J(a_s, b_s) &= -2 \sum_{i=1}^{N} y_s(k) \left(y_s(k+1) - a_s y_s(k) - b_s w_s(k)\right) = 0, \\ \frac{\partial}{\partial b_s} J(a_s, b_s) &= -2 \sum_{i=1}^{N} w_s(k) \left(y_s(k+1) - a_s y_s(k) - b_s w_s(k)\right) = 0. \end{aligned} \tag{C.11}$$

The solution to the system of equations is:

$$\begin{aligned} a_s &= \frac{S_3S_4 - S_2S_5}{S_1S_3 - S_2^2}, \\ b_s &= \frac{S_1S_5 - S_2S_4}{S_1S_3 - S_2^2}, \end{aligned} \tag{C.12}$$

where:

$$\begin{aligned} S_1 &= \sum_{k=1}^{N} y_s^2(k), \\ S_2 &= \sum_{k=1}^{N} w_s(k)y_s(k), \\ S_3 &= \sum_{k=1}^{N} w_s^2(k), \\ S_4 &= \sum_{k=1}^{N} y_s(k)y_s(k+1), \\ S_5 &= \sum_{k=1}^{N} w_s(k)y_s(k+1). \end{aligned} \tag{C.13}$$

There are three methods to determine the accuracy of the model. The first metric is the *root-mean-square error* (RMSE), defined as:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{k=1}^{N} e^2(k+1)}. \tag{C.14}$$

The second metric is the *variability* of the model, denoted by R^2 and defined as:

$$R^2 = 1 - \frac{\text{Var}(e(k)}{\text{Var}(y(k))}. \tag{C.15}$$

The variability ranges from 0 to 1 and a value of 1 indicates a perfect model fit.

The third metric is the *correlation coefficient* (CC), defined as:

$$\text{CC} = \frac{\sum_{k=1}^{N} e(k)w_s(k)}{\sqrt{\text{Var}(e(k))\text{Var}(w_s(k))}}. \tag{C.16}$$

A smaller correlation coefficient indicates a better fit.