Kwaliteitsgedreven beheer van videodiensten
in segmentgebaseerde cachenetwerken

Quality-Driven Management of Video Streaming Services
in Segment-Based Cache Networks

Maxim Claeys

Promotoren: prof. dr. ir. F. De Turck, prof. dr. S. Latré
Proefschrift ingediend tot het behalen van de graden van
Doctor in de ingenieurswetenschappen: computerwetenschappen (Universiteit Gent) en
Doctor in de wetenschappen: informatica (Universiteit Antwerpen)

UNIVERSITEIT
GENT

Vakgroep Informatietechnologie
Voorzitter: prof. dr. ir. B. Dhoedt
Faculteit Ingenieurswetenschappen en Architectuur

Universiteit
Antwerpen

Departement Wiskunde en Informatica
Voorzitter: prof. dr. C. Blondia
Faculteit Wetenschappen

Academiejaar 2016 - 2017

ISBN 978-90-8578-981-9
NUR 986, 988
Wettelijk depot: D/2017/10.500/16

UNIVERSITEIT
GENT

Universiteit Gent
Faculteit Ingenieurswetenschappen en Architectuur
Vakgroep Informatietechnologie

Universiteit
Antwerpen

Universiteit Antwerpen
Faculteit Wetenschappen
Departement Wiskunde en Informatica

Leden van de examencommissie:

prof. dr. ir. Filip De Turck (promotor)
Universiteit Gent - imec
prof. dr. Steven Latré (promotor)
Universiteit Antwerpen - imec

prof. dr. ir. Rik Van de Walle (voorzitter)
Universiteit Gent - imec
prof. dr. Chris Blondia (co-voorzitter)
Universiteit Antwerpen - imec
prof. dr. Peter Lambert (secretaris)
Universiteit Gent - imec
ir. Werner Van Leekwijck
Nokia Bell Labs
dr. ir. Daphné Tuncer
University College London
prof. dr. Tobias Hoßfeld
Universität Duisburg-Essen
prof. dr. ir. Danny De Vleeschauwer
Nokia Bell Labs - Universiteit Gent

iwt

Dit werk kwam tot stand in het kader van een
specialisatiebeurs van het IWT-Vlaanderen
(Instituut voor de aanmoediging van Innovatie door
Wetenschap en Technologie in Vlaanderen)

Proefschrift tot het behalen van de graden van
Doctor in de ingenieurswetenschappen:
computerwetenschappen (Universiteit Gent) en
Doctor in de wetenschappen:
informatica (Universiteit Antwerpen)
Academiejaar 2016-2017

Dankwoord

Woensdag 1 augustus 2012. Ik herinner me mijn eerste werkdag bij IBCN alsof het gisteren was. Met een klein hartje aankomen in de Zuiderpoort, de laptop ophalen bij de admins en wat bedeesd kennis maken met de nieuwe collega's van bureau 2.21. Het is nauwelijks te bevatten dat we ondertussen zowat 4,5 jaar verder zijn en dat de tijd is aangebroken om een van mijn laatste taken als doctorandus aan te vatten: het schrijven van het dankwoord van mijn doctoraatsboek. Ik ben dan ook erg dankbaar te kunnen terugblikken op een fantastisch avontuur waarbij ik heel wat heb bijgeleerd, niet alleen op wetenschappelijk vlak, maar zeker ook op persoonlijk vlak.

Eerst en vooral wil ik Filip De Turck hiervoor bedanken. Zonder hem was ik hoogstwaarschijnlijk nooit aan dit doctoraat begonnen. Bedankt Filip voor alle kansen en de tonnen vertrouwen die je me hebt gegeven. Bedankt voor je open deur wanneer ik nog maar eens de inhoud van een paper wou bespreken of wanneer een nieuwe onderzoeksrichting moest worden bepaald. De invalshoeken die je telkens opnieuw bood waren een verrijking voor dit doctoraatsonderzoek en ik heb er door de jaren heen erg veel uit geleerd. Ik had het geluk om op mijn eerste dag bij IBCN onder de vleugels van Steven Latré terecht te komen. Wanneer ik terugkijk op die eerste maanden, besef ik dat mij begeleiden voor hem meer dan een full-time job moet geweest zijn. Ik kan het dan ook goed begrijpen dat hij na anderhalf jaar naar Antwerpen is gevlucht. In deze context wil ik zeker ook Jeroen Famaey bedanken om deze leegte in eerste instantie op uitmuntende wijze in te vullen (om kort daarna weliswaar dezelfde vluchtroute te volgen). Bedankt Steven om me zo intensief te begeleiden tijdens mijn eerste stappen in de onderzoekswereld en om ook na je ontsnappingspoging nog steeds tijd vrij te maken in je drukke agenda om feedback te geven op elke paper of revisie. Bedankt Filip en Steven voor de vele interessante besprekingen, maar zeker ook voor de talrijke informele babbels. Het was een echte eer om ruim 4 jaar lang intensief te mogen samenwerken met twee top-promotoren zoals jullie.

Het spreekt voor zich dat bij een doctoraat heel wat meer komt kijken dan onderzoek alleen. De omkadering binnen IBCN is er echter een waar veel onderzoeksgroepen ongetwijfeld een puntje kunnen aan zuigen. Veel bewondering gaat dan ook uit naar Piet Demeester om een forse onderzoeksgroep als deze op uitstekende wijze te leiden. Het kloppend hart van IBCN bevindt zich zonder twijfel in het secretariaat bij Martine en Davinia. Bedankt om elke vraag met de glimlach te beantwoorden en me steeds bij te staan met raad en daad voor elke administratieve beslommering. Maar bovenal bedankt voor de vele gezellige praatjes om

de dag steeds goed mee te beginnen. Jullie zijn van goudwaarde voor deze onderzoeksgroep! Ook op vlak van financiële administratie en technische ondersteuning worden we steeds in de watten gelegd door Bernadette, Joke en het A-team (Joeri, Bert, Simon, Vicent en Brecht). Ook een dikke merci aan Sabrina om onze dagelijkse werkomgeving proper te houden en voor de warme begroeting elke ochtend. Naast alle mensen bij IBCN wil ik ook graag het instituut voor de aanmoediging van Innovatie door Wetenschap en Technologie in Vlaanderen (IWT) bedanken om 4 jaar lang in mij te investeren. Dit liet me toe om mijn onderzoek uit te voeren zonder me zorgen te moeten maken over contractverlengingen of andere financiële rompslomp.

De warme collegiale sfeer binnen IBCN werd reeds in talloze voorgaande dankwoorden geroemd, maar ook voor mij zorgde die ervoor dat ik elke dag met plezier naar de Zuiderpoort of iGent kwam en dat work-at-home absoluut niks voor mij was. In de eerste plaats wil ik daarvoor de (ex-)bureaugenoten van 2.21 en 200.012 bedanken. Bedankt Bram, Jeroen S, Kristof, Leandro, Merlijn, Niels, Olivier, Philip, Piet, Sander, Stefano, Steven B, Steven VC, Thijs, Thomas V, Tim V en Wim om van onze bureau een toffe werkplek te maken met een juiste balans tussen de nodige job-ernst, informele babbels en grappen en grollen. Bedankt om al die tijd te kunnen leven met mijn fratsen, voor de gezellige lunchpauzes en de sporadische activiteiten buiten de kantooruren. Het doet me veel plezier te zien dat je ook na enkele reorganisaties op vlak van bureauindeling op je collega's kan rekenen. Bedankt Tim V en Steven B om jullie robots even aan de kant te laten staan om mij te helpen met moeilijke leesverslagen. Fijne collega's vind je natuurlijk ook over de bureaugrenzen heen. Bedankt Bart, Bruno, Femke en Femke, Jeroen vdH, Pieter, Pieter-Jan, Stijn, Thomas D, Tim W en Wannes voor de vlotte samenwerking tijdens verschillende project of onderwijsactiviteiten en voor de luchtige babbels aan de koffiemachine.

Na verloop van tijd leer je ook de mens achter je collega's kennen. Het doet dan ook plezier dat sommigen onder hen zonder dat je het merkt uitgroeien van collega's tot vrienden. Bedankt Jeroen S voor het memorabele avondje uit in het exotisch Maldegem. Bedankt Thomas V voor de vele geslaagde edities van Tomorrowland die we samen achter de rug hebben en voor de ontelbare andere topmomenten die we mochten beleven. Bedankt dat ik altijd op jou kon terugvallen, ook tijdens je drukke Qrama verhaal.

Ook al heb ik ontelbare mooie momenten meegemaakt op de Zuiderpoort en in iGent, de mooiste herinneringen van een doctorandus worden gemaakt tijdens internationale conferenties of projectmeetings. Bedankt aan de zowat vaste reisgezellen Filip, Hendrik, Jeroen F, Niels, Pieter-Jan, Stefano en Steven L. Ik zal nog vaak met plezier terugdenken aan het Oktoberfest in Munchen, de match in het legandarische Maracanã in Rio de Janeiro, de wanhopige masseuses op Barceloneta, de heerlijke midnight-snack in Krakow, de onverwachte regenbui tijdens de wandeling (of was het een survival-tocht?) in Algonquin provincial park in Canada, de ontmoeting met de enige echte Doo Doo The Clown in Ottawa, de hopeloze zoektocht naar een bar in het "bruisende" Montreal, de biertram in Brno, en ga zo maar door. Het waren stuk voor stuk ervaringen die ik nooit zal vergeten.

In elk van bovenstaande passages kon ik Niels Bouten vermeld hebben, maar sommige mensen verdienen nu eenmaal een eigen paragraaf. Het is geen toeval dat een groot deel van de bovenstaande herinneringen ook in jouw dankwoord stonden vermeld, want sinds de eerste dag dat we samen op het MISTRAL project mochten werken, werden we snel twee handen op één buik. Ik leerde je kennen als een top-onderzoeker, maar ik zal me je altijd herinneren als een top-vriend. Bedankt voor de buitengewone samenwerking tijdens onze vele gemeenschappelijke projecten, bedankt voor de ontelbare mooie herinneringen op conferentie, bedankt voor de peptalk wanneer het even nodig was, maar bovenal bedankt om elke dag opnieuw met de glimlach naar mijn zever te luisteren. Ik heb aan dit doctoraat een fantastische vriend overgehouden, en dat alleen al maakte het meer dan de moeite waard.

Het leven bestaat uiteraard uit meer dan doctoraatsonderzoek alleen. Daarom wil ik graag mijn familie en schoonfamilie bedanken. Bedankt meter, meme en pepe, Thomas, Katrien en Dirk, Erna en Armand, Céline en Glenn voor alle warme familiemomenten waarbij alle zorgen die nu en dan bij een doctoraat komen kijken even vergeten kunnen worden. Dat er zo nog veel mogen volgen! In het bijzonder bedankt aan mijn lieve ouders voor de warme thuis, voor alle kansen die jullie me hebben geboden en nog steeds bieden. Bedankt om altijd in mij te geloven, ook wanneer ik dat zelf niet deed. Geluk schuilt vaak in kleine dingen: dankjewel voor de kilo's verse fruitsla tijdens 'den blok' en de motiverende schouderklop wanneer het even nodig was. Zonder jullie onvoorwaardelijke steun had ik hier nu niet gestaan. Bedankt Yaël voor de hilarische sing-alongs op weg van of naar het station en om elke examenperiode opnieuw een motivatie geweest te zijn. Ik kijk meer op naar mijn kleine zus dan je zou denken. Met jullie drieën hebben jullie van het ouderlijk huis steeds een warme thuis gemaakt. Bedankt lieve kleine Noé voor de motiverende 'dikke duim' voor de interne verdediging.

Tot slot wil ik de grootste liefde in mijn leven bedanken. Liefste Eva, tijdens de afgelopen 4,5 jaar zijn vele zaken veranderd, maar de onafscheidelijke band die we hebben is daar geen van. Bedankt om me te steunen in alles wat ik doe en er altijd voor mij te zijn. Bedankt mollie om me elke dag opnieuw gelukkig te maken en me te tonen wat echt belangrijk is in het leven. Ik ben ontzettend dankbaar dat ik elke dag van dit doctoraat met jou heb kunnen delen. 2017 mag dan wel het einde betekenen van dit avontuur, het betekent ook het begin van het belangrijkste en allermooiste verhaal uit ons leven. Binnen enkele maanden mag ik je eindelijk mijn vrouw noemen en dat vervult mijn hart met trots. Ik kan niet wachten om de rest van mijn leven met jou door te brengen. Lieve schat, ik hou van jou.

Gent, februari 2017
Maxim Claeys

Table of Contents

List of Figures

List of Tables

List of Acronyms

0-9

3GPP 3rd Generation Partnership Project

A

AIMD Additive Increase/Multiplicative Decrease
ARC Adaptive Replacement Cache

C

CAPEX Capital Expenditures
CCN Content-Centric Networking
CDN Content Delivery Network
CDNI Content Delivery Network Interconnection

D

DASH Dynamic Adaptive Streaming over HTTP
DVD Digital Versatile Disc

E

ECN Explicit Congestion Notification
EDF Earliest Deadline First

eMOS estimated Mean Opinion Score
EWMA Exponentially Weighted Moving Average

F

FAQ-Learning Frequency Adjusted Q-Learning

G

GPS Global Positioning System

H

HAS HTTP Adaptive Streaming
HLS HTTP Live Streaming
HTTP Hypertext Transfer Protocol

I

ICN Information-Centric Networking
IETF Internet Engineering Task Force
ILP Integer Linear Program
IP Internet Protocol
IPTV Internet Protocol television
ISP Internet Service Provider

L

LCE Leave Copy Everywhere
LFU Least Frequently Used
LFU-DA Least Frequently Used (LFU) Dynamic Aging
LRU Least Recently Used

M

MOS Mean Opinion Score
MPEG Moving Picture Experts Group
MSS Microsoft ISS Smooth Streaming

N

NFV Network Function Virtualization

O

OTT Over-The-Top

P

P2P Peer-to-Peer
PoP Point of Presence

Q

QoE Quality of Experience
QoS Quality of Service
QUIC Quick UDP Internet Connections

R

RED Random Early Detection
RL Reinforcement Learning
RTP Real-time Transport Protocol
RTCP RTP Control Protocol

RTSP Real-Time Streaming Protocol
RTT Round-Trip Time

S

SCAP Shared Content Addressing Protocol
SDN Software-Defined Networking

T

telco-CDN telco-Content Delivery Network
TCP Transmission Control Protocol
TLS Transport Layer Security

U

UDP User Datagram Protocol

V

VDBE Value-Difference Based Exploration
VDBES Value-Difference Based Exploration with Softmax action selection
VoD Video-on-Demand
VoIP Voice over IP

W

WWW World Wide Web

Samenvatting
– Summary in Dutch –

Tijdens de afgelopen decennia is het internet geëvolueerd van een onderzoeksnetwerk met enkele honderden knopen tot een netwerk dat miljarden gebruikers over de hele wereld met elkaar verbindt. Deze verbluffende evolutie werd aangestuurd door een onafgebroken vooruitgang op vlak van aangeboden diensten, gebruikte technologie en gebruikersverwachtingen. Waar het internet in de begindagen hoofdzakelijk gebruikt werd voor statische diensten zoals e-mail en surfen op het web, wordt vandaag een brede waaier van hoofdzakelijk multimediadiensten aangeboden. Zo vertegenwoordigen videodiensten momenteel meer dan 70% van het wereldwijde internetverkeer. Bovendien wordt verwacht dat de populariteit van deze diensten de komende jaren alleen maar zal toenemen. Niet alleen het dienstenaanbod is sterk uitgebreid, maar tegenwoordig worden ook uiteenlopende toestellen gebruikt om deze diensten te gebruiken, gaande van desktop computers en laptops tot slimme televisies, smartphones en tablet computers. Daarbovenop nemen de gebruikersverwachtingen over de aangeboden diensten, bijvoorbeeld op vlak van beeldkwaliteit en reactiesnelheid, voortdurend toe. De sleutel tot het succes van videodiensten schuilt dan ook in het efficiënt invullen van deze hoge verwachtingen. Daarom is de zogenaamde Quality of Experience (QoE), een maatstaf voor de gebruikerservaring, een belangrijke waardemeter bij het beheer van multimediadiensten.

Ook de technologieën die worden gebruikt om videodiensten af te leveren zijn doorheen de tijd sterk geëvolueerd. Terwijl oorspronkelijk gespecialiseerde protocollen werden aangewend, hebben Hypertext Transfer Protocol (HTTP)-gebaseerde technieken de laatste jaren heel wat aan populariteit gewonnen. Deze technieken hebben enkele onmiskenbare voordelen zoals de betrouwbare aflevering over Transmission Control Protocol (TCP), de mogelijkheid tot hergebruik van standaard caches en de moeiteloze integratie met firewalls. Om in te spelen op dynamische netwerkomstandigheden werd HTTP Adaptive Streaming (HAS) voorgesteld. In HAS wordt de video opgesplitst in verschillende segmenten van enkele seconden. Bovendien wordt elk van deze segmenten in verschillende kwaliteitsniveaus aangeboden. In de videospeler wordt een heuristiek voorzien die voor elk segment het meest geschikte kwaliteitsniveau selecteert op basis van de waargenomen netwerkomstandigheden en de huidige buffervulling. De laatste jaren is HAS de de facto standaard geworden voor Over-The-Top (OTT) videodiensten. Gezien de vele voordelen van deze techniek geniet HAS recent ook steeds meer aandacht in

scenario's waarbij het netwerk wordt beheerd door de operator. Ondanks de adaptiviteit van HAS kunnen kwaliteitsschommelingen en onderbrekingen in de video nog steeds voorkomen. Dit heeft vanzelfsprekend een negatieve invloed op de gebruikerservaring. In het verleden werden kwaliteitsdegradaties voorkomen door het netwerk sterk te overdimensioneren. Gezien de hoge populariteit van videodiensten en de toenemende gebruikersverwachtingen is dit economisch niet langer haalbaar. Daarom is het efficiënt beheren van videodiensten van essentieel belang geworden om aan de hoge kwaliteitsvereisten te kunnen voldoen en tegelijkertijd de druk op de onderliggende netwerkinfrastructuur onder controle te houden. In deze thesis worden dan ook verschillende beheersoplossingen voorgesteld die de aflevering van HTTP-gebaseerde videodiensten optimaliseren. Deze optimalisaties richten zich zowel op de serverzijde, de gebruikerszijde als op aanpassingen in het netwerk.

Een eerste manier om de geleverde diensten te verbeteren is het optimaliseren van de kwaliteitsselectie in de videospeler. Ook al werden de HAS-protocollen gestandaardiseerd door MPEG Dynamic Adaptive Streaming over HTTP (DASH), toch is een grote hoeveelheid aan commerciële kwaliteitsselectieheuristieken beschikbaar. De huidige implementaties zijn echter deterministisch en toegespitst op specifieke netwerkomstandigheden. Hierdoor zijn ze minder geschikt om met een brede waaier aan dynamische netwerkomstandigheden om te gaan. De huidige kwaliteitsselecties zijn dan ook vaak niet optimaal waardoor een aanzienlijk deel van de videostromen invloed ondervindt van lage kwaliteit en onderbrekingen. Daarom werd een zelflerende heuristiek ontwikkeld. Door gebruik te maken van een techniek gebaseerd op Reinforcement Learning (RL) zal de videospeler zijn gedrag voortdurend aanpassen aan de waargenomen netwerkomstandigheden en de huidige buffervulling. In deze thesis wordt aangetoond dat deze heuristiek huidige deterministische algoritmes sterk kan overtreffen op vlak van QoE in dynamische netwerkomstandigheden.

Naast aanpassingen aan de videospeler kunnen ook optimalisaties in het netwerk worden toegepast om de aflevering van videodiensten efficiënter te maken en de druk op de netwerkinfrastructuur te verminderen. Daarom zijn Internet Service Providers (ISPs) recentelijk begonnen met het uitrollen van zogenaamde telco-Content Delivery Networks (telco-CDNs). Dit biedt hen meer controle over hun infrastructuur door de videos in hun eigen netwerk dichter bij de eindgebruiker te kunnen cachen. Bovendien stelt de huidige vooruitgang op vlak van netwerkvirtualisatie hen in staat om hun telco-CDN infrastructuur te virtualiseren. Dit laat hen toe om op een dynamische manier virtuele opslag- en afleveringsdiensten aan te bieden aan verschillende partijen. Gebaseerd op dit scenario werd een hybride caching strategie voorgesteld die de allocatie van opslagcapaciteit, het plaatsen van data en het selecteren van een server optimaliseert over deze verschillende partijen heen. Door het periodiek proactief plaatsen op basis van populariteitsvoorspellingen te combineren met klassieke reactieve caching, kan de druk op het ISP netwerk sterk worden verlicht. Tegelijkertijd worden de videos dichter bij de eindgebruiker gebracht, wat dan weer de QoE ten goede komt.

De prestatie van een reactieve caching aanpak is sterk afhankelijk van de ge-

bruikte vervangingsstrategie. Aangezien de videosegmenten met HAS worden afgeleverd via HTTP kunnen traditionele algoritmes, gebruikt bij web caching, eenvoudig worden hergebruikt. De eigenschappen van gesegmenteerde video kunnen echter worden aangewend om de prestaties van de caches sterk te verbeteren. Er werden dan ook algoritmes ontwikkeld die de temporele structuur in een gesegmenteerde video in rekening brengen bij cachevervangingen. In deze thesis werd hier op verder gebouwd door rekening te houden met het feit dat steeds meer gebruikers van videodiensten verschillende afleveringen van dezelfde serie na elkaar bekijken. Naar dit fenomeen wordt vaak verwezen onder de noemer *binge watching*. Gebaseerd op deze trend werd een strategie ontwikkeld die aanneemt dat wanneer een gebruiker een aflevering van een bepaalde serie bekijkt, hij vervolgens de daaropvolgende aflevering zal bekijken. Door deze informatie in rekening te brengen kan de efficiëntie van de cachenetwerken sterk verhoogd worden, wat leidt tot een vermindering van de druk op de onderliggende netwerkinfrastructuur.

Als laatste optimalisatie werd voorgesteld om deadlines geassocieerd met videodiensten in rekening te brengen in de transportlaag van het netwerk. Aangezien ze worden afgeleverd over TCP, concurreren HTTP-gebaseerde videodiensten om de beschikbare bandbreedte met andere diensten met uiteenlopende karakteristieken. Elke byte in een videostroom kan echter worden geassocieerd met een impliciete deadline. Wanneer het videosegment niet wordt afgeleverd vooraleer het moet worden afgespeeld, zal een onderbreking optreden. Aangezien dit een negatieve invloed heeft op de gebruikerservaring werd een deadline-gebaseerd adaptatiealgoritme voorgesteld op basis van een geparametriseerde versie van TCP New Reno. Deze aanpak vereist op de transportlaag enkel aanpassingen aan de serverzijde en is dan ook volledig transparant voor de rest van het netwerk. Er werd aangetoond dat de QoE van een HAS videodienst sterk verbeterd kan worden door het dynamisch aanpassen van de aggressiviteit van een TCP videostroom.

De oplossingen die worden voorgesteld in deze thesis behandelen enkele belangrijke uitdagingen die momenteel optreden in HTTP-gebaseerde gesegmenteerde videodiensten door de gebruikerservaring te verbeteren en tegelijkertijd de druk op de netwerkinfrastructuur te beperken. Verder onderzoek kan hierop verderbouwen door te blijven focussen op nieuwe evoluties, zowel op technologisch vlak als op vlak van gebruikersverwachtingen. Zo biedt de huidige interesse in nieuwe protocollen zoals HTTP/2 bijvoorbeeld een aantal nieuwe mogelijkheden. Door de toenemende tendens van dergelijke protocollen richting in-netwerk encryptie gaat dit echter ook gepaard met een aantal belangrijke uitdagingen op vlak van in-netwerk optimalisaties. Bovendien zorgen de sterke diversificatie van gebruikerstoestellen en hun verhoogde mobiliteit voor toenemende gebruikersverwachtingen. Tenslotte introduceren nieuwe netwerkstrategieën zoals Software-Defined Networking (SDN) en Network Function Virtualization (NFV) een aantal nieuwe scenario's die de flexibiliteit van videodiensten sterk kunnen doen toenemen. De sterke interactie tussen dergelijke dynamische netwerkbeslissingen en de efficiëntie van strategieën zoals proactieve plaatsing van data zorgt er echter voor dat dit een uitdagende opdracht wordt.

Summary

Over the last decades, the Internet has evolved from a research network connecting a couple of hundred nodes to a system connecting billions of users around the globe. This astonishing evolution has been driven by a continuous innovation, both in terms of technology, service offering and user requirements. While in the early days the Internet was mainly used for static services such as e-mail and web browsing, a wide variety of rich services is offered today, mostly focusing on multimedia delivery. More specifically, video streaming services are dominating the Internet today, representing over 70% of the total consumer Internet traffic worldwide. Furthermore, this popularity increase is projected to continue in the years to come. Not only the number of services has grown, but nowadays a wide variety of devices are used to access these services as well, ranging from desktop computers and laptops to connected TVs, smartphones and tablets. Furthermore, the end-user requirements are continuously increasing in terms of minimum resolution, response time, etc. The key for a video streaming service to be successful is to meet these stringent quality requirements. Therefore, the quality as experienced by the end-user, denoted as Quality of Experience (QoE), is an important metric when it comes to the management of multimedia delivery services.

Over time, the technologies used to deliver video streaming services have been subjected to change as well. While originally dedicated streaming protocols have been used, Hypertext Transfer Protocol (HTTP)-based streaming methods have gained a lot of popularity over the last years. This delivery type comes with several important advantages, such as the reliable transmission over the Transmission Control Protocol (TCP), the reuse of standard proxies and caches and the seamless integration with firewalls. To cope with dynamic network conditions, HTTP Adaptive Streaming (HAS) has been proposed. In HAS, the video is temporally split in multiple segments, each of which are encoded at different quality levels. At the client side, a rate adaption heuristic is applied to change the requested quality level based on, for example, the perceived network conditions, the buffer filling level and the device characteristics. In latest years, HAS has become the de facto standard for Over-The-Top (OTT) video streaming. Given the strong advantages of this approach, recently a lot of interest is gained in managed delivery scenarios as well. Despite the adaptive capabilities of HAS, due to the best effort nature of the delivery, quality oscillations and video freezes can still occur, negatively impacting the QoE. Historically, network resources have been strongly over-provisioned to avoid quality degradations. However, given the popularity of video streaming services and the increasingly stringent quality requirements, this

is no longer economically viable. Therefore, efficiently managing the delivery of video streaming services becomes of utmost importance to provide high QoE to the end-users while simultaneously keeping the strain imposed on the underlying network resources under control. In this dissertation, multiple management solutions are proposed to optimize the end-to-end delivery of HTTP-based streaming services, both in terms of QoE and resource efficiency. The proposed approaches include server side, user side as well as in-network optimizations.

A first way to improve the delivered service quality is to optimize the quality selection process at the client side. Even though the interfaces and protocol data of HAS are standardized in MPEG Dynamic Adaptive Streaming over HTTP (DASH), a vast amount of commercial rate adaptation heuristics exist. However, state-of-the-art implementations are deterministic and hardwired to specific network configurations, making them unable to cope with a vast range of highly dynamic network environments. As a result, current rate adaptation decisions are often sub-optimal, regularly causing HAS streaming sessions to suffer from video freezing and low resolution video. Therefore, in this thesis, a self-learning quality selection heuristic is proposed. By using a Reinforcement Learning (RL)-based technique, the HAS client continuously adapts its behavior based on the perceived network characteristics and buffer filling level. It is shown in this dissertation that this self-learning approach can significantly outperform current deterministic algorithms in terms of QoE in dynamic network environments.

Besides client side approaches, in-network optimizations can be applied to increase the efficiency of the service delivery and to reduce the load on the underlying network resources. To restrain the pressure exerted on their networks by current delivery strategies, Internet Service Providers (ISPs) have started to deploy so-called telco-Content Delivery Networks (telco-CDNs). This gives them more control over their resources by allowing content to be cached deep inside the ISP network. Furthermore, current advances in network virtualization enable ISPs to virtualize their telco-CDN infrastructure, allowing them to dynamically offer virtual storage and content delivery services to multiple third parties. Based on this scenario, a hybrid cache management strategy has been proposed, optimizing the capacity allocation, content placement and server selection across multiple tenants. By combining periodical proactive content placements based on popularity predictions with reactive caching capacity, the load on the ISP network can be reduced while simultaneously bringing the content closer to the end-user, positively influencing the QoE.

The performance of a reactive caching approach strongly depends on the design of the cache replacement algorithms. As in HAS the video segments are delivered over HTTP, regular web caching algorithms can easily be reused. However, the characteristics of segmented video contain important information that can be exploited to improve the performance of the cache network. Therefore, cache replacement algorithms have been proposed, taking into account the temporal structure of a segmented video stream. This dissertation builds on this work by considering the fact that recently, a significant amount of users frequently watch multiple consecutive episodes of the same series in one sitting. This phenomenon is com-

monly referred to as binge watching. Based on this trend, an announcement-based caching strategy is proposed. By assuming that when a user is watching episodic content, the following episode will be streamed after the current one, the hit ratio of the caching network can be significantly increased, resulting in a reduced load on the underlying network resources.

A final optimization approach proposed in this dissertation focuses on bringing deadline-awareness to the transport layer. Being delivered over TCP, HTTP-based streaming services compete for bandwidth with other types of services with different characteristics, following the fair-share paradigm. However, every byte in a video stream has an implicit strict deadline. If the data is not delivered to the client before it is needed to be played out, the video freezes, detrimentally impacting the QoE. Therefore, a deadline-based adaptation algorithm is proposed, based on a parametrized version of TCP New Reno congestion control. This approach only requires server-side adaptations at the transport layer and consequently is fully transparent for the rest of the network. It was shown that by taking into account the deadline information at the transport layer to dynamically adapt the aggressiveness of the TCP video stream, the QoE can significantly be increased in an HAS Video-on-Demand (VoD) scenario.

The proposed approaches address important challenges currently perceived in segment-based video streaming services over HTTP by improving the quality delivered to the end-user while simultaneously restraining the load exposed on the underlying network infrastructure. Future research can build on this work by further focusing on new evolutions, both in terms of technology and user requirements. For example, the current adoption of new protocols such as HTTP/2 and their trend towards in-network encryption offers some interesting opportunities as well as important challenges when it comes to in-network service optimizations. Furthermore, the diversification of client devices and the increasing mobility of the end-users add to the requirements for video streaming services. Finally, novel networking paradigms such as Software-Defined Networking (SDN) and Network Function Virtualization (NFV) open up some new scenarios to add flexibility to the service delivery. However, the strong interplay between dynamic networking decisions and application layer approaches such as content placement and caching make this a challenging task.

1 Introduction

"Quality is never an accident. It is always the result of intelligent effort."

–John Ruskin (1819 - 1900)

1.1 The Internet video streaming evolution

October 29, 1969, 10:30PM. UCLA student Charley Kline sends the first successful message over a packet switched network between the University of California in Los Angeles and Stanford University in Palo Alto [1]. ARPANET was born. In the years to follow, ARPANET grew slow but steady to a network with over 200 host computers by the early 80s. With the introduction of the World Wide Web (WWW) in 1990, Tim Berners-Lee triggered an astonishing evolution that brought the Internet to what it is today, connecting over 3.6 billion users worldwide. Over the past decades, this evolution was driven by a continuous introduction of new services.

While originally the Internet was mainly used for e-mail and web browsing, a wide variety of rich services is offered today, ranging from social media (e.g., Facebook, Twitter) to video streaming services (e.g., YouTube, Netflix). This shift towards multimedia delivery started with the introduction of Peer-to-Peer (P2P) file sharing services such as Napster, Gnutella and Kazaa in the late 90s, allowing users to easily share large files. Even though Microsoft and RealNetworks pioneered in streaming technologies over packet networks as early as the mid 90s, multimedia was usually delivered over non-streaming channels (i.e. large

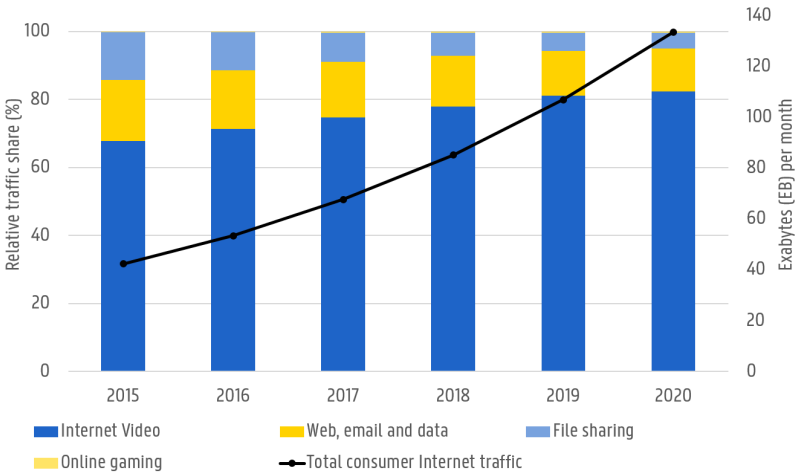


Figure 1.1: Forecast of the global consumer Internet traffic between 2015 and 2020 [2].

file transfers) due to the limited network capacities at that time. When connection speeds significantly increased in the early 2000s, streaming services gained popularity, demonstrated by the success of YouTube, initiated in 2005. While YouTube targeted users sharing short home-made videos, other players such as content providers and telecommunication companies gained interest in Internet video streaming technologies as well. As an example, Netflix originally started in the DVD by mail business, but soon introduced their Video-on-Demand (VoD) streaming service in February 2007, offering movies and series in high quality, which is now their dominant service offering. The popularity of video streaming services skyrocketed and has been dominating the Internet market since 2010. As shown in Figure 1.1, this trend is projected to continue in the years to come, with Internet video expected to represent more than 80% of the global consumer Internet traffic by 2020 [2]. Furthermore, the quality requirements for video streaming services are continuously increasing as well. While in the early days of YouTube very low quality videos were commonly accepted, less than high definition video is no longer sufficient to satisfy the end-user today.

Video streaming services can generally be subdivided in two categories: Internet Protocol television (IPTV) and Over-The-Top (OTT). IPTV services are commonly offered by network operators in combination with telephony and Internet access. The delivered content mostly covers live television broadcast as well as a limited VoD catalog. As the video streams are delivered over their managed networks, operators can provide delivery guarantees, hence ensuring a minimum level of quality. However, this commonly limits the delivery to the customer premises

through a set-top box. On the contrary, OTT video streaming is delivered over the traditional Internet, allowing the content to be consumed on a wide variety of devices at any location with Internet connectivity. On the downside, due to the best-effort nature of the Internet, no hard delivery guarantees can be offered, often leading to quality degradations as well. Nevertheless, the increasing popularity of OTT video streaming services has put operators under increasing pressure to offer additional services to their customers, allowing them to access supplementary content or from different locations and other devices than the set-top box. For this purpose, traditional IPTV providers have started to deploy OTT technologies as well [3].

Obviously, the technologies used to deliver OTT streaming services have been subjected to an evolutionary process as well. Traditionally, Real-time Transport Protocol (RTP) in conjunction with RTP Control Protocol (RTCP) has been commonly used as an application layer streaming protocol. While the media itself is transported using RTP over User Datagram Protocol (UDP), RTCP is used to monitor transmission statistics. Real-Time Streaming Protocol (RTSP) was often used on top to control the media sessions between client and server. However, as this is a stateful protocol, the scalability is strongly impacted. For several years, the video streaming industry is steadily shifting away from these classic streaming protocols, back to Hypertext Transfer Protocol (HTTP) over Transmission Control Protocol (TCP). This trend was mostly initiated by the fact that classic streaming protocols often have difficulty getting around firewalls. Using plain HTTP, such problems do not occur. Furthermore, HTTP-based streaming services are delivered reliably over TCP and can reuse standard proxies and caches.

In the most basic form of video delivery over HTTP, the multimedia file is treated as a regular file and downloaded over HTTP. As this requires the entire video file to be downloaded before it can be played, this results in a significant startup delay. Therefore, HTTP progressive download was quickly introduced. With progressive download, the video file is temporally split into multiple segments containing a couple of seconds of video each. At the server side, a meta data file is used to link the different segments into a single video. Based on this information, the client progressively downloads the subsequent segments, allowing the playback to start as soon as a couple of segments are available. However, when the download speed is not sufficient to stay ahead of the playback, continuous buffering occurs, negatively impacting the user experience. For this reason, HTTP Adaptive Streaming (HAS) has been proposed to cope with dynamic network conditions. In HAS, the video is not only temporally segmented, but also qualitatively, all segments are encoded at different quality levels. At the client side, the network statistics and the buffer filling level are monitored. Based on this information, the requested quality for each segment can be defined. In this way, HAS can avoid buffer depletions by reducing the video quality when the available bandwidth is

limited. As this rate adaptation logic is fully distributed at the client side, regular stateless HTTP servers can be used, strongly benefiting scalability. Several large industrial players have commercial implementations of the HAS concept, including Microsoft ISS Smooth Streaming (MSS) [4], HTTP Live Streaming (HLS) by Apple [5] and Adobe's HTTP Dynamic Streaming [6]. In 2011, the Moving Picture Experts Group (MPEG) tried to find common ground between the vast amount of commercial implementations by standardizing the interfaces and protocol data in Dynamic Adaptive Streaming over HTTP (DASH) [7]. In the latest years, HAS has been adopted by important players such as YouTube and Netflix, and quickly became the de facto standard for video streaming, representing more than 60% of the consumer Internet traffic in North America [8].

To efficiently deliver streaming services to a broad audience, globally distributed networks of proxy servers, called Content Delivery Networks (CDNs), are commonly used. CDN nodes are usually deployed in multiple data centers spread across different locations to increase the global availability of the content. By doing so, the load on the origin server is significantly decreased and the content is brought closer to the end-user, positively impacting the user experience. Furthermore, content providers started to build their own CDNs by collaborating with Internet Service Providers (ISPs) to optimize the delivery of their services by placing dedicated servers inside the providers' networks (e.g., Netflix Open-Connect [9]). Doing so, content providers can benefit from their knowledge about the usage of their service to optimize content management decisions. However, these content management operations are usually performed with no or very limited information about the ISP networks (e.g., topology, resource utilization, etc.) they are interacting with. This lack of knowledge can lead to inefficient resource allocation, exerting an immense strain on the ISP's resources and exposing them to large Capital Expenditures (CAPEX) [10]. To address this issue, ISPs started to deploy so called telco-Content Delivery Networks (telco-CDNs), increasing the control over their network resources while simultaneously optimizing the delivered quality to their end-users [11].

1.2 Problem statement

As presented in the previous section, the differences between the technologies used to deliver managed and OTT services are fading away, both shifting to HTTP-based techniques. However, alongside the technological evolutions, the quality expectations imposed by the end-users of video streaming services continuously increase as well (e.g., 4K video, 3D video, different camera angles, time-shifting, etc.). As a result, the providers of both managed and OTT video streaming services are faced with the important challenge to provide high Quality of Experience (QoE) to the end-users while simultaneously keeping the strain imposed on

the underlying network resources under control. Given the excessive popularity of video streaming services, in the future this can no longer be achieved by over-provisioning the network as was done in the latest decades. Therefore, efficiently managing the delivery of video streaming services becomes of utmost importance. In this dissertation, multiple management solutions are proposed, optimizing the service delivery at the client side, at the server side, as well as in the network in between. More specifically, the following problems are identified and tackled in this thesis:

1. *Existing HAS rate adaptation heuristics are tailored to specific network configurations.* As introduced in the previous section, HAS techniques have gained a lot of popularity thanks to their ability to cope with bandwidth fluctuations and network congestion. A client side rate adaptation heuristic is used to monitor the current device and network characteristics and to adapt the downloaded video quality appropriately. While MPEG, in collaboration with other standard groups, such as 3rd Generation Partnership Project (3GPP), standardized the HAS interfaces in DASH in 2011, the adaptation heuristics are still implementation specific. Given the increasing popularity of mobile devices for video streaming services, these heuristics are commonly used with different access technologies and corresponding network characteristics. However, state-of-the-art quality selection heuristics are deterministic and tailored to specific network configurations. Therefore, they are unable to cope with a vast range of highly dynamic network environments. As a result, current rate adaptation decisions and corresponding QoE are often sub-optimal. This claim is supported by Conviva, showing that in 2014, 28.8% and 58.4% of the HAS streaming sessions suffered from video freezing and low resolution video, respectively [12].

2. *Current video delivery over CDNs exerts a lot of pressure on ISP networks.* In latest years, video streaming services heavily rely on CDNs to deliver the content in a scalable way. Furthermore, large content providers such as Netflix started to deploy their own CDN infrastructure. To meet the increasingly stringent quality requirements, CDN providers currently aim at bringing the content even closer to the end-users in order to reduce both the latency and the bandwidth consumption. A common way to achieve this goal is to physically place dedicated servers inside the ISP network or connect them to a nearby Internet exchange point, through manually-negotiated contractual agreements. However, this delivery strategy exerts an immense strain on the underlying ISP network resources. As an example, with the introduction of Netflix in Belgium in September 2014, Telenet reported that the video streaming service accounted for 10% of the total Internet traffic during the opening weekend. Similar claims have been made by other ISPs

in other countries, with every new Netflix launch. As this results in increasing operating costs and decreasing revenues for the ISPs, they have started to explore alternative business models and service offerings. This has lead to the deployment of telco-CDNs which allow content to be cached deep inside the ISP network. Even though this gives ISPs more control over their resources, current content placement algorithms are not optimized for multi-tenant scenarios with dynamic capacity allocations.

3. *The management of video streaming services does not fully take into account the changing user behavior.* Over the last decades, not only the video streaming technology has changed, but the way these services are used has evolved as well. While with the introduction of YouTube the main focus of VoD streaming was on short home-made videos, VoD services like Netflix and Hulu are now commonly used to stream entire movies and series. These services make it easy for the end-users to watch as much as they want, whenever they want, stimulating the consumers' desire to organize their own schedule, rather than following a fixed timetable as with linear television. This has lead to the phenomenon of *binge watching*, where users consecutively watch multiple episodes of the same series in one sitting, to become the everyday behavior. A study on user behavior by Nielsen has shown that in 2013, respectively 88% and 70% of Netflix and Hulu Plus users regularly stream three or more episodes of the same TV show consecutively [13]. A survey on binge watching, performed by Conviva in 2015, has shown that on average, 2.3 episodes of a series are watched in a single sitting [14]. Furthermore, as presented in Figure 1.2, the results from this survey show that users commonly binge-watch multiple series simultaneously. These user behavior trends contain valuable information that could potentially be used to optimize the service delivery by anticipating expected video requests. However, current streaming solutions focus on single stream delivery, without considering dependencies between consecutive streams.

4. *Fair bandwidth sharing introduced by TCP is far from minimizing the number of deadline misses.* The popularity of HTTP-based streaming techniques is mainly due to the reliable delivery over TCP and the seamless interaction with existing network equipment such as firewall. As multiple flows compete for bandwidth, TCP uses a congestion control strategy to avoid congestive collapse. It was shown that by using such congestion control strategy, multiple flows competing for bandwidth in comparable environments eventually converge to using equal amounts of a contended link. However, a wide variety of services with different characteristics and requirements are delivered over TCP. For example, each byte in a video stream has an implicit strict deadline. If the data is not delivered at the client before this

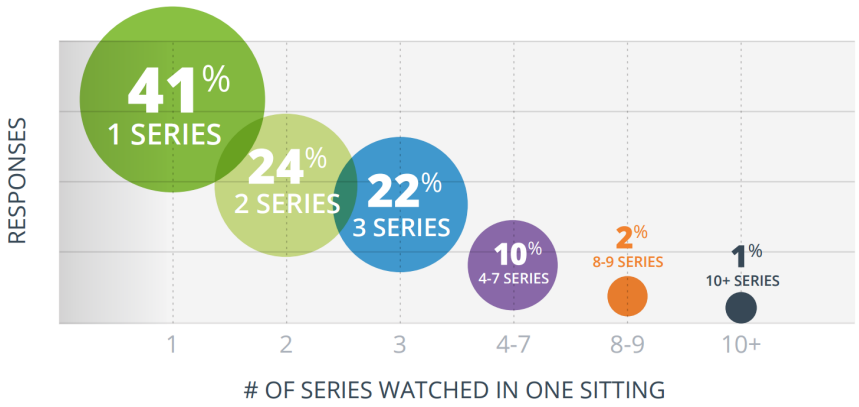


Figure 1.2: Overview of the binge watching behavior according to Conviva [14].

deadline, the video playout is interrupted, detrimentally impacting the QoE. Other types of services (e.g., email, filetransfer, ...) have no strict timing constraints. It is clear that fair bandwidth sharing between flows with different requirements might not be optimal. Even though HAS offers a lot of advantages, the delivery over TCP using current congestion control strategies is not well suited to guarantee a continuous video stream.

1.3 Dissertation outline

This dissertation is composed of a number of publications that were obtained within the scope of this PhD. These selected publications provide an integral and consistent overview of the realized work. The different research contributions are detailed in Section 1.4, while Section 1.5 presents the complete list of publications that resulted from this work. In this section, an overview of the remainder of this dissertation is presented. Figure 1.3 gives an overview of the different contributions that are presented in each chapter to optimize the video streaming service delivery at different locations in the network.

Chapter 2 focuses on the client-side optimization of OTT video delivery by introducing a self-learning HAS client. As opposed to state-of-the art quality selection heuristics, which are deterministic and tailored to specific network configurations, this Reinforcement Learning (RL)-based adaptation algorithm is able to dynamically adapt its behavior to the perceived networking environment in order to optimize the QoE. Furthermore, extensions to the basic self-learning client are proposed to significantly decrease the learning time and to strongly improve the performance in variable environments. Thorough evaluations showed that using this approach, the QoE can be significantly increased compared to state-of-the-art

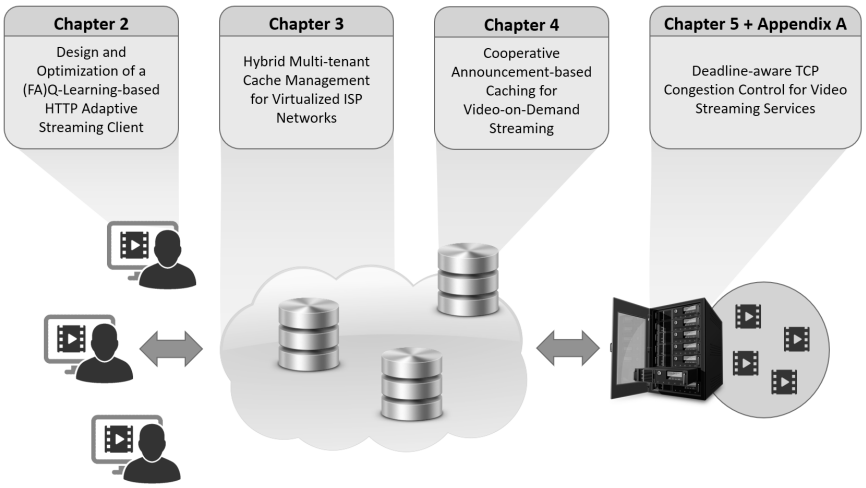


Figure 1.3: Overview of this PhD dissertation.

heuristics in a wide variety of network environments.

In-network optimizations are presented in Chapter 3 and Chapter 4. Chapter 3 considers the scenario where ISPs deploy telco-CDNs to reduce the load on their network resources. Virtualization of storage and networking resources can enable the ISP to simultaneously lease its telco-CDN infrastructure to multiple third parties, opening up new business models and revenue streams. A hybrid cache management approach is proposed where proactive content placement and traditional reactive cache replacement are combined. A proactive algorithm optimizes the allocation of storage capacity and the content placement decisions based on predicted content popularity and the geographical distribution of requests. By simultaneously allocating caching capacity applying reactive cache replacement strategy strategies, reactivity to unexpected changes in the request pattern is provided. It was shown that this cache management strategy can significantly reduce the bandwidth usage inside the ISP network while simultaneously bringing the content closer to the end-user, positively influencing the QoE.

When reactive caching is considered, designing appropriate replacement strategies is of utmost importance to achieve high caching efficiency and effectively reduce the network load. Therefore, algorithms have been introduced to optimize caching of segmented video by taking into account the temporal structure of the video. In Chapter 4, two cache replacement strategies are proposed to additionally take advantage of the binge watching phenomenon to further reduce the network load. Given this phenomenon, when a user watches an episode of a specific series, it can be assumed that the next episode will be requested subsequently. Using this additional information, future reuse times of the segmented content can be

estimated for consecutive episodes. In this way, the caching efficiency can be significantly increased compared to algorithms only applying reuse time predictions based on the structure of a single segmented video. The two proposed approaches differ in the level of cooperation inside the caching network. While a first approach applies a simple threshold-based cascading strategy to provide a basic level of coordination, the second proposal relies on a more advanced election-based coordination strategy.

In a video stream, hard deadlines are associated with each packet. In general, client applications are aware of the deadlines associated with the requested content. We show that multimedia delivery can be strongly optimized by prioritizing specific flows when these deadlines are introduced in the network. In Chapter 5, a deadline-aware TCP congestion control strategy is proposed, taking into account the deadline information to change the aggressiveness of a stream. This proposal only requires changes at the server side and reduces to standard TCP congestion control when no deadline information is available. Therefore, this approach is fully transparent to the network. Evaluations have shown that using the proposed congestion control strategy can significantly reduce the bottleneck bandwidth required to deliver TCP-based video streams without video freezes. Appendix A elaborates on this approach and provides evaluations in a VoD HAS scenario, allowing to assess the impact on the QoE.

1.4 Research contributions

This dissertation aims to reduce the strain imposed on network resources while simultaneously improving the QoE of HTTP-based segmented video streaming services. The main contributions made in this area can be identified as follows:

1. A self-learning rate adaptation heuristic to deal with a wide variety of network configurations. (Chapter 2)
    - Design of a Q-learning-based HAS rate adaptation heuristic to dynamically learn the best actions corresponding to the actual network environment.
    - A Frequency Adjusted Q-Learning (FAQ-Learning) extension of the self-learning algorithm to increase the performance when operating in strongly variable environments.
    - An estimation algorithm to incorporate HAS domain knowledge into the initial Q-tables in order to boost the client performance during the learning phase.
    - A QoE estimation model, combining state-of-the-art estimation models for different quality aspects such as average video quality, switching behavior and video freezing.

- Thorough evaluations using an NS-3-based simulation framework, demonstrating the increased QoE in variable bandwidth configurations compared to the MSS algorithm.

2. A hybrid cache management strategy for virtualized telco-CDNs to reduce the load on ISP network resources. (Chapter 3)

   - An Integer Linear Program (ILP) formulation of the multi-tenant content placement and server selection problem, taking into account the content migration overhead.
   - A basic popularity prediction strategy for VoD requests, combining short- and long-term historical data.
   - Design of a cache division strategy, balancing proactive content placement and reactive cache replacement.
   - Thorough evaluations using a real request trace of the VoD service of a major European ISP, showing the performance of the proposed approach in terms of both network and caching metrics.
   - Analysis of the achievable performance increase when augmenting the popularity prediction with exogenous information about the future request pattern, e.g., based on trends in social media.

3. An announcement-based cache replacement strategy taking advantage of the binge watching user behavior to reduce the load on the network resources. (Chapter 4)

   - Design of a cache replacement algorithm taking advantage of knowledge about future segment requests when streaming segmented video and additionally exploiting the added knowledge about expected future episode requests.
   - Two coordination strategies within the caching network. The first approach applies a simple threshold-based cascading strategy while the second proposal adds a more advanced election-based strategy.
   - A detailed analysis of the influence of user behavior in terms of video interruptions on both the proposed and state-of-the-art caching strategies.
   - Thorough evaluations in a distributed caching scenario on multiple topologies to assess the performance of the proposed approaches, both in terms of network and caching metrics.

4. A deadline-aware TCP congestion control strategy, optimizing video streaming delivery. (Chapter 5 & Appendix A)

- A parametrization of the congestion avoidance phase of the TCP New Reno congestion control mechanism.
- A feasibility study of prioritizing TCP flows by changing the configuration of these parameters.
- Design of algorithms to dynamically adapt the parameter configuration of streams to change the aggressiveness based on the deadline information.
- Thorough evaluations using large scale packet-based simulation in NS-3, showing the possible bottleneck bandwidth reduction for TCP-based video streaming.
- Analysis of the impact on the QoE in an HAS VoD scenario using large scale simulations in NS-3.

1.5 Publications

The research results obtained during this PhD research have been published in scientific journals and presented at a series of international conferences. The following list provides an overview of the publications during my PhD research.

1.5.1 A1: Journal publications indexed by the ISI Web of Science "Science Citation Index Expanded"

1. Daphne Tuncer, Vasilis Sourlas, Marinos Charalambides, **Maxim Claeys**, Jeroen Famaey, George Pavlou, Filip De Turck. *Scalable Cache Management for ISP-operated Content Delivery Services.* Published in Journal of Selected Areas in Communications (JSAC), IEEE, Volume 34, Issue 8, Pages 2063-2076, August 2016. doi:10.1109/jsac.2016.2577319.

2. **Maxim Claeys**, Daphne Tuncer, Jeroen Famaey, Marinos Charalambides, Steven Latré, George Pavlou, Filip De Turck. *Hybrid Multi-tenant Cache Management for Virtualized ISP Networks.* Published in Journal of Network and Computer Applications (JNCA), Elsevier, Volume 68, Pages 28-41, June 2016. doi:10.1016/j.jnca.2016.04.004.

3. **Maxim Claeys**, Niels Bouten, Danny De Vleeschauwer, Werner Van Leekwijck, Steven Latré, Filip De Turck. *Cooperative Announcement-based Caching for Video-on-Demand Streaming.* Published in Transactions on Network and Service Management (TNSM), IEEE, Volume 13, Issue 2, Pages 308-321, June 2016. doi:10.1109/tnsm.2016.2546459.

4. Stefano Petrangeli, Jeroen Famaey, **Maxim Claeys**, Filip De Turck, Steven Latré. *QoE-driven Rate Adaptation Heuristic for Fair Adaptive Video Streaming.* Published in Transactions on Multimedia Computing, Communications and Applications (TOMM), ACM, Volume 12, Issue 2, Pages 28:1-28:24, March 2016. doi:10.1145/2818361.

5. Nicolas Staelens, Jonas De Meulenaere, **Maxim Claeys**, Glenn Van Wallendael, Wendy Van den Broeck, Jan De Cock, Filip De Turck, Rik Van de Walle, Piet Demeester. *Subjective Quality Assessment of Longer Duration Video Sequences Delivered over HTTP Adaptive Streaming to Tablet Devices.* Published in Transactions on Broadcasting, IEEE, Volume 60, Issue 4, Pages 707-714, December 2014. doi:10.1109/tbc.2014.2359255.

6. **Maxim Claeys**, Steven Latré, Jeroen Famaey, Filip De Turck. *Design and Evaluation of a Self-Learning HTTP Adaptive Video Streaming Client.* Published in Communications Letters, IEEE, Volume 18, Issue 4, Pages 716-719, April 2014. doi:10.1109/lcomm.2014.020414.132649.

7. **Maxim Claeys**, Steven Latré, Jeroen Famaey, Tingyao Wu, Werner Van Leekwijck, Filip De Turck. *Design and Optimization of a (FA)Q-Learning-based HTTP Adaptive Streaming Client.* Published in Connection Science, Taylor & Francis, Volume 26, Issue 1, Pages 25-43, March 2014. doi:10.1080/09540091.2014.885273.

8. Femke Ongenae, **Maxim Claeys**, Wannes Kerckhove, Thomas Dupont, Piet Verhoeve, Filip De Turck. *A Self-learning Nurse Call System.* Published in Computers in Biology and Medicine, Elsevier, Volume 44, Pages 110-123, January 2014. doi:10.1016/j.compbiomed.2013.10.014.

9. Femke Ongenae, **Maxim Claeys**, Thomas Dupont, Wannes Kerckhove, Piet Verhoeve, Tom Dhaene, Filip De Turck. *A Probabilistic Ontology-based Platform for Self-learning Context-aware Healthcare Applications.* Published in Expert Systems with Applications, Elsevier, Volume 40, Issue 18, Pages 7629-7646, December 2013. doi:10.1016/j.eswa.2013.07.038.

1.5.2 P1: Proceedings included in the ISI Web of Science "Conference Proceedings Citation Index - Science"

1. Niels Bouten, **Maxim Claeys**, Rashid Mijumbi, Joan Serrat, Jeroen Famaey, Steven Latré. *Semantic Validation of Affinity Constrained Service Function Chain Requests.* In proceedings of the IEEE Conference on Network Softwarization (NetSoft), Seoul, Korea, Pages 202-210, June 2016. doi:10.1109/netsoft.2016.7502414.

2. **Maxim Claeys**, Niels Bouten, Danny De Vleeschauwer, Werner Van Leekwijck, Steven Latré, Filip De Turck. *An Announcement-based Caching Approach for Video-on-Demand Streaming.* In proceedings of the International Conference on Network and Service Management (CNSM), Barcelona, Spain, Pages 310-317, November 2015. doi:10.1109/cnsm.2015.7367376.

3. Stefano Petrangeli, Niels Bouten, **Maxim Claeys**, Filip De Turck. *Towards SVC-based Adaptive Streaming in Information-centric Networks.* In proceedings of the IEEE International Conference on Multimedia & Expo Workshops (ICMEW), Torino, Italy, Pages 1-6, July 2015. doi:10.1109/icmew.2015.7169859.

4. Jeroen van der Hooft, Stefano Petrangeli, **Maxim Claeys**, Jeroen Famaey, Filip De Turck. *A Learning-based Algorithm for Improved Bandwidth Awareness of Adaptive Streaming Clients.* In proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM), Ottawa, Canada, Pages 131-138, May 2015. doi:10.1109/inm.2015.7140285.

5. **Maxim Claeys**, Daphne Tuncer, Jeroen Famaey, Marinos Charalambides, Steven Latré, George Pavlou, Filip De Turck. *Proactive Multi-tenant Cache Management for Virtualized ISP Networks.* In proceedings of the International Conference on Network and Service Management (CNSM), Rio de Janeiro, Brazil, Pages 82-90, November 2014. doi:10.1109/cnsm.2014.7014144.

6. **Maxim Claeys**, Daphne Tuncer, Jeroen Famaey, Marinos Charalambides, Steven Latré, Filip De Turck, George Pavlou. *Towards Multi-tenant Cache Management for ISP Networks.* In proceedings of the European Conference on Networks and Communications (EuCNC), Bologna, Italy, Pages 1-5, June 2014. doi:10.1109/eucnc.2014.6882692.

7. Rashid Mijumbi, Juan-Luis Gorricho, Joan Serrat, **Maxim Claeys**, Jeroen Famaey, Filip De Turck. *Neural Network-based Autonomous Allocation of Resources in Virtual Networks.* In proceedings of the European Conference on Networks and Communications (EuCNC), Bologna, Italy, Pages 1-6, June 2014. doi:10.1109/eucnc.2014.6882668.

8. **Maxim Claeys**, Steven Latré, Filip De Turck. *Efficient Management of Virtualized Information-Centric Networks.* In proceedings of the International Conference on Autonomous Infrastructure, Management and Security (AIMS), Brno, Czech Republic, Pages 42-46, June 2014. doi:10.1007/978-3-662-43862-6_4.

9. Niels Bouten, **Maxim Claeys**, Robin Bailleul, David Lou, Jeroen Famaey, Steven Latré, Jan De Cock, Filip De Turck, Werner Van Leekwijck.

Improved Delivery of Live SVC-based HTTP Adaptive Streaming Content. In proceedings of the IEEE Network Operations and Management Symposium (NOMS), Krakow, Poland, Pages 1-2, May 2014. doi:10.1109/noms.2014.6838269.

10. Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, **Maxim Claeys**, Filip De Turck, Steven Latré. *Design and Evaluation of Learning Algorithms for Dynamic Resource Management in Virtual Networks.* In proceedings of the IEEE Network Operations and Management Symposium (NOMS), Krakow, Poland, Pages 1-9, May 2014. doi:10.1109/noms.2014.6838258.

11. Stefano Petrangeli, **Maxim Claeys**, Steven Latré, Jeroen Famaey, Filip De Turck. *A Multi-Agent Q-Learning-based Framework for Achieving Fairness in HTTP Adaptive Streaming.* In proceedings of the IEEE Network Operations and Management Symposium (NOMS), Krakow, Poland, Pages 1-9, May 2014. doi:10.1109/noms.2014.6838245.

12. Niels Bouten, **Maxim Claeys**, Steven Latré, Jeroen Famaey, Werner Van Leekwijck, Filip De Turck. *Deadline-based Approach for Improving Delivery of SVC-based HTTP Adaptive Streaming Content.* In proceedings of the IEEE Network Operations and Management Symposium (NOMS), Krakow, Poland, Pages 1-7, May 2014. doi:10.1109/noms.2014.6838402.

13. Sebastiaan Laga, Thomas Van Cleemput, Filip Van Raemdonk, Felix Vanhoutte, Niels Bouten, **Maxim Claeys**, Filip De Turck. *Optimizing Scalable Video Delivery Through OpenFlow Layer-based Routing.* In proceedings of the IEEE Network Operations and Management Symposium (NOMS), Krakow, Poland, Pages 1-4, May 2014. doi:10.1109/noms.2014.6838378.

1.5.3 C1: Other publications in international conferences

1. **Maxim Claeys**, Niels Bouten, Danny De Vleeschauwer, Koen De Schepper, Werner Van Leekwijck, Steven Latré, Filip De Turck. *Deadline-aware TCP Congestion Control for Video Streaming Services.* In proceedings of the International Conference on Network and Service Management (CNSM), Montreal, Canada, Pages 100-108, October 2016. doi:10.1109/cnsm.2016.7818405.

2. Niels Bouten, **Maxim Claeys**, Bert Van Poecke, Steven Latré, Filip De Turck. *Dynamic Server Selection Strategy for Multi-server HTTP Adaptive Streaming Services.* In proceedings of the International Conference on Network and Service Management (CNSM), Montreal, Canada, Pages 82-90, October 2016. doi:10.1109/cnsm.2016.7818403.

3. Stefano Petrangeli, **Maxim Claeys**, Filip De Turck, Tim Wauters, Patrick Van Staey. *Energy-aware Quality Adaptation for Mobile Video Streaming.* In proceedings of the International Conference on Network and Service Management (CNSM), Montreal, Canada, Pages 253-257, October 2016. doi:10.1109/cnsm.2016.7818427.

4. **Maxim Claeys**, Steven Latré, Jeroen Famaey, Tingyao Wu, Werner Van Leekwijck, Filip De Turck. *Design of a Q-Learning-based Client Quality Selection Algorithm for HTTP Adaptive Video Streaming.* In proceedings of the Workshop on Adaptive and Learning Agents (ALA), Saint Paul (MN), USA, Pages 30-37, May 2013.

References

[1] B. Leiner, V. Cerf, D. Clark, R. Kahn, L. Kleinrock, D. Lynch, J. Postel, L. Roberts, and S. Wolff. *Brief History of the Internet*. ACM SIGCOMM Computer Communication Review, 39(5):22–31, 2009.

[2] Cisco. *Cisco Visual Networking Index: Forecast and methodology, 2015-2020*. Technical report, Cisco, 2016.

[3] DASH-FI. *Survey of European broadcasters on MPEG-DASH*. Technical report, DASH Industry Forum, 2013.

[4] Microsoft. *Smooth Streaming: The Official Microsoft IIS Site*, 2008. Available from: https://www.iis.net/downloads/microsoft/smooth-streaming.

[5] Apple. *HTTP Live Streaming*, 2016. Available from: https://tools.ietf.org/html/draft-pantos-http-live-streaming-20.

[6] Adobe. *HTTP Dynamic Streaming*, 2009. Available from: http://www.adobe.com/products/hds-dynamic-streaming.html.

[7] T. Stockhammer. *Dynamic adaptive streaming over HTTP: Standards and design principles*. In Proceedings of the ACM International Conference on Multimedia Systems, pages 133–144, 2011.

[8] Sandvine. *Global Internet Phenomena*. Technical report, Sandvine - Intelligent Broadband Networks, 2016.

[9] Netflix. *How Netflix Works With ISPs Around the Globe to Deliver a Great Viewing Experience*, 2016. Available from: https://media.netflix.com/en/company-blog/how-netflix-works-with-isps-around-the-globe-to-deliver-a-great-viewing-experience.

[10] CellularNews. *Overall telecom CAPEX to rise in 2011 due to video, 3G, LTE investments*, 2010. Available from: http://www.cellular-news.com/story/46986.php.

[11] D. Tuncer, M. Charalambides, R. Landa, and G. Pavlou. *More control over network resources: an ISP caching perspective*. In Proceedings of the International Conference on Network and Service Management (CNSM), pages 26–33, 2013.

[12] Conviva. *2015 viewer experience report*. Technical report, Conviva, 2015.

[13] Nielsen. *"Binging" is the new viewing for over-the-top streamers*. Technical report, Nielsen, 2013.

[14] Conviva. *Binge watching: the new currency of video economics*. Technical report, Conviva, 2015.

2

Design and Optimization of a (FA)Q-Learning-based HTTP Adaptive Streaming Client

M. Claeys, S. Latré, J. Famaey, T. Wu, W. Van Leekwijck and F. De Turck.

Published in Connection Science, March 2014.

⋆ ⋆ ⋆

This chapter focuses on the problem that current HTTP Adaptive Streaming (HAS) rate adaptation heuristics are hardwired to fit specific network configuration. Therefore, they are less flexible to fit a vast range of network conditions. To tackle this problem, a (Frequency Adjusted)Q-Learning HAS client is proposed. In contrast to existing heuristics, the proposed HAS client dynamically learns the optimal behavior corresponding to the current network environment in order to optimize the Quality of Experience (QoE). Furthermore, the client has been optimized both in terms of global performance and convergence speed. Thorough evaluations show that the proposed client can outperform deterministic algorithms by 11% to 18% in terms of Mean Opinion Score (MOS) in a wide range of network configurations.

2.1 Introduction

Over the past decades, multimedia services have gained a lot of popularity. This growth is largely due to video streaming services. These services can generally be divided into Internet Protocol television (IPTV), offered by a network provider and managed through resource reservation, and Over-The-Top (OTT) services, streamed over a network provider's network without his intervention (e.g. YouTube[1] and Netflix[2]). HTTP Adaptive Streaming (HAS) techniques are becoming the de-facto standard for OTT video streaming. Large industrial players such as Microsoft, Apple and Adobe have commercial implementations of the HAS concept available. These HTTP-based techniques split video content into small segments of typically 2s to 10s, encoded at multiple quality levels. This approach allows video clients to dynamically adapt the requested video quality to fit the perceived network state. Based on the perceived characteristics, such as delay and throughput, a quality selection heuristic is used at the client side to determine the quality level to request for the next segment, in order to maximize the Quality of Experience (QoE).

HAS comes with important advantages. Not only is the video content delivered reliably over HTTP, HAS also allows seamless interaction through firewalls. On the downside, delivery over the best-effort Internet makes these techniques prone to network congestion and large bandwidth fluctuations due to cross traffic, which can be detrimental for the QoE, the quality as perceived by the end-users. HAS client behavior is therefore a crucial factor for the streaming service to be beneficial and to ensure a sufficient level of QoE for the end-user.

Current HAS client heuristics are however hard-coded to fit specific network configurations. This makes current approaches less flexible to deal with a vast range of network setups and corresponding bandwidth variations. Even though they are well suited for a specific network configuration, these deterministic approaches yield unsatisfactory results when the environment changes. This chapter proposes a Reinforcement Learning (RL) based HAS client, allowing dynamic adjustment of streaming behavior to the perceived network state. RL is a machine learning technique, designed to operate in situations in which an agent only has limited knowledge about the environment, leading to a high degree of uncertainty concerning how the environment will react to the performed actions. However, interaction with the environment is the only way for the agent to learn. At each state in the environment, the agent perceives a numerical reward, providing feedback to the agent's actions. The agent's goal is to learn which action to take in a given state of the environment, in order to maximize the cumulative numerical reward [1]. Mapping this RL principle to the HAS scenario, the agent learns which

[1] http://www.youtube.com
[2] http://www.netflix.com

quality level to request in the perceived network state.

The contributions of this chapter are three-fold. First, a Q-Learning-based HAS client has been designed. This approach, in contrast to traditional heuristics, allows the client to dynamically learn the best actions corresponding to the actual network environment. Second, a Frequency Adjusted Q-Learning (FAQ-Learning) approach is proposed to increase the client performance in strongly variable environments. Third, an estimation algorithm is presented to incorporate HAS domain knowledge into the initial Q-Tables in order to boost the client performance during the learning phase. All of the presented approaches are thoroughly evaluated using a network-based video streaming simulation framework. The simulation results allow comparison with the Microsoft ISS Smooth Streaming algorithm, of which the original source code is available.

The remainder of this chapter is structured as follows. First, the basic HAS principle is discussed in Section 2.2. Next, Section 2.3 gives an overview of related work, both on HAS and RL. Section 2.4 elaborates on the design of the proposed self-learning HAS client. Next to a general overview, this section presents the applied RL techniques, exploration policies and the constructed environmental state and reward function. Furthermore, in Section 2.5, the initial Q-Table estimation algorithm is proposed. The evaluations of the presented self-learning HAS client are described in Section 2.6. Finally, Section 2.7 presents some final conclusions.

2.2 HTTP Adaptive Streaming

HAS is the third generation of HTTP based streaming and is increasingly being used in OTT video delivery. Several large industrial players have commercial implementations of the HAS concept, including Microsoft ISS Smooth Streaming (MSS)[3], HTTP Live Streaming (HLS) by Apple[4] and Adobe's HTTP Dynamic Streaming[5]. In 2011, MPEG established the common ground between the vast amount of commercial implementations by standardizing the interfaces and protocol data in Dynamic Adaptive Streaming over HTTP (DASH) [2]. The bit rate adaptation heuristics are, however, not standardized, and thus implementation specific.

Regardless of heuristic details, all of these implementations follow the general HAS concept, shown in Figure 2.1. In HAS, a video consists of multiple segments with a typical length of 2s to 10s, encoded at multiple quality levels. At the client side, a manifest file, containing information about the segments and quality levels, is used to link the different segments into a single video stream. Based on the information in the manifest file, the HAS client sequentially requests the next seg-

[3]http://www.iis.net/downloads/microsoft/smooth-streaming
[4]http://tools.ietf.org/html/draft-pantos-http-live-streaming-10
[5]http://www.adobe.com/products/hds-dynamic-streaming.html

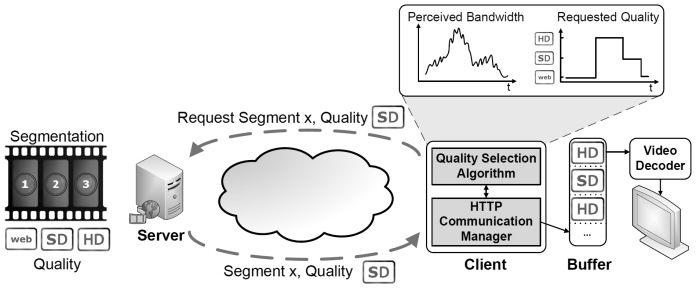


Figure 2.1: Schematic overview of the HAS concept.

ment upon arrival of the previous segment. Based on the network state, perceived while downloading previous segments, a quality selection heuristic dynamically adapts the requested quality level in order to optimize the QoE. Each segment is downloaded in a progressive manner, while a buffer at the client side is used to take care of temporary anomalies such as a late arrival of a video segment. Finally, the video segments, stored in the buffer, are played back as a single continuous video stream. Current quality adaptation algorithms for HAS are deterministic and tailored to specific network configurations, hampering the ability to react to a vast range of highly dynamic network settings. On the contrary, this chapter proposes a self-learning HAS client to autonomously react to changing network conditions, as will be discussed in Section 2.4.

2.3 Related work

2.3.1 HAS client algorithms

As described in Section 2.2, multiple proprietary HAS algorithms are available. *Akhshabi et al.* compare several commercial and open source HAS clients and identify their inefficiencies, such as excessive quality switching. Recently, several new client approaches have been described in literature [3]. *Liu et al.* propose a client heuristic to handle parallel HTTP connections, based on the segment download time [4]. By comparing the perceived segment download time with the expected segment download time, bandwidth fluctuations can be estimated appropriately. The opportunities of HAS in the domain of live streaming services are investigated by *Lohmar et al.* [5]. The work focuses on the influence of client buffering on the end-to-end delay. Many recent research topics focus on the applicability of HAS in mobile environments by exploiting additional information. The heuristic described by *Riiser et al.* uses Global Positioning System (GPS) information to obtain more accurate information on the available bandwidth [6]. Furthermore,

Adzic et al. have proposed content-aware heuristics [7]. These approaches require meta-data to be embedded in the video description. The additional consequences of quality selection in mobile environments have been shown by *Trestian et al.* [8]. The research shows that lowering the requested quality can significantly reduce energy consumption of Android devices. *Jarnikov et al.* discuss several guidelines on configuring robust HAS clients with regard to changing network conditions [9]. The authors model the quality selection problem as a Markov Decision Process (MDP) which is solved offline. The performance is only guaranteed when the resulting strategy is applied in the same network environment as was modeled in the MDP. The same rationale holds for the dynamic programming approach, proposed by *Xiang et al.*, where the bandwidth transition probabilities are needed to calculate the client policy offline [10].

In contrast to the above described approaches, we focus on an increased adaptivity and self-learning behavior of the client heuristic through the design of a RL-based client approach. An initial approach to the application of a self-learning agent for HAS has been presented in previous work [11]. Even though the presented approach showed promising results, further experiments have shown that the performance in a variable networking environment was unsatisfactory. Therefore, the client has been thoroughly redesigned from the ground by reducing the environmental state space and reward definition. Furthermore, this chapter presents several techniques to further boost the performance and multiple network configurations have been evaluated. The use of a RL agent in HAS clients has also been proposed by *Menkovski et al.*, applying the SARSA(λ) technique [12]. Even though convergence of the learning agent is shown, a general evaluation of the client is infeasible since no comparison to existing approaches is provided.

2.3.2 Learning in adaptive streaming

Even though learning has not been applied frequently in the area of HAS, multiple RL-based adaptive streaming techniques have been proposed in the literature. Where our self-learning HAS approach is focused on the client side, existing RL-based adaptive streaming techniques target server or network side solutions to Quality of Service (QoS) provisioning for adaptive streaming systems. *Fei et al.* formulate call admission control and bandwidth adaptation for adaptive multimedia delivery in mobile communication networks as a Markov Decision Problem (MDP), which they solve using Q-Learning [13]. RL is applied by *Charvillat et al.* to create a dynamic adaptation agent, considering both user behavior and context information. Furthermore, this generic approach is applied to solve a ubiquitous streaming problem [14]. Artificial neural networks are used by *McClary et al.* to dynamically adapt the audio transmission rate in mobile ad-hoc networks, considering available throughput, end-to-end delay and jitter [15].

2.3.3 Learning in QoS/QoE optimization

Reinforcement Learning (RL) has previously been successfully applied to various network management problems. *Cao* proposes an agent-based network fault diagnosis model in which the agent uses RL to improve its fault diagnosis performance [16]. Their research shows this approach can outperform traditional fault diagnosis models. *Bagnasco et al.* propose the application of RL to dynamically adapt a hierarchical policy model to perform autonomous network management [17]. They argue that an autonomic system must have a degree of flexibility to adapt to changes in goals or resources, which is hard to achieve by means of static policies. In the area of resource allocation, successful applications of RL can be found. *Vengerov* presents a general framework for adaptive reconfiguration of distributed systems using a combination of RL and fuzzy rule-bases [18]. Dynamic resource allocation of entities sharing a set of resources is used as an example. On the other hand, *Tesauro et al.* propose a hybrid approach, gaining performance in the combination of RL and deterministic queuing models for resource allocation [19]. In this hybrid system, RL is used to train offline on collected data, hereby avoiding possible performance loss during the online training phase. Furthermore, multiple approaches have been proposed, focusing on the resource allocation aspect in wireless mesh networks [20, 21]. Another area of network management where RL has been applied previously is QoS routing. Especially in wireless sensor networks, the network topology may change frequently, yielding inherently imprecise state information, which impedes QoS routing. *Ouferhat et al.* propose a Q-Learning based formalism to optimize QoS scheduling [22]. *Parakh et al.* propose a decentralized bandwidth allocation for video streams in wireless systems, based on game theory [23]. In this system, users are charged for bandwidth resources proportionally to the requested bit-rate. *Mastronarde et al.* apply RL to the problem of energy-efficient point-to-point transmission of delay-sensitive (multimedia)data over a wireless communication channel [24].

2.4 Reinforcement learning-based HAS client

2.4.1 Approach

As discussed in Section 2.2, current deterministic HAS clients only have limited abilities to react to a vast range of dynamic network settings. We propose the usage of a learning agent to enable the HAS client to adapt its behavior by interacting with the network environment. In this way, the client will be able to react to network conditions that were not under consideration when designing the typical deterministic quality selection algorithms.

2.4.2 Q-Learning

A commonly used RL algorithm is Q-Learning [25]. Using Q-Learning, knowledge regarding both reward prospects and environmental state transitions are obtained through interaction with the environment. In Q-Learning, Q-values $Q(s,a)$ are used to measure the "Quality" of taking a specific action a in a certain state s, based on the perceived rewards. By applying eligibility traces [25], current rewards are not only credited to taking the last action, but also to actions taken further in the past. Therefore, additional variables $e(s,a)$ are introduced for every state-action pair (s,a), indicating the degree to which taking action a in state s is *eligible* for undergoing learning changes when a new reward is perceived.

Equations (2.1) and (2.2) respectively show how the eligibility traces and the Q-values are updated when action a is taken in state s, yielding a reward r and new state s'. In these equations, (s,a) is the state-action pair and $\alpha \in [0;1]$ and $\gamma \in [0;1]$ are the learning rate and the discount factor respectively. The parameter λ is referred to as the trace-decay parameter. I_{xy} denotes an identity indicator function, equal to 1 if $x = y$ and 0 otherwise.

$$e(x,y) = I_{xs} \cdot I_{ya} + \begin{cases} \lambda\gamma e(x,y) & : Q(s,a) = \max_b Q(s,b) \\ 0 & : \text{else} \end{cases} \tag{2.1}$$

$$Q(s,a) = Q(s,a) + \alpha e(s,a)\left[r + \gamma \max_b Q(s',b) - Q(s,a)\right] \tag{2.2}$$

The action to be performed in a specific state is selected based on the learned Q-values. The specific selection tactic depends on the used policy (see Section 2.4.4).

2.4.3 Frequency Adjusted Q-Learning

One problem that occurred using Q-Learning in preliminary simulations is the slow reaction to changes in the environment. When an action has a significantly lower Q-value than another action for a specific state, it has very low probability to be selected. When the environment changes, the new information on the quality of the action is only obtained very slowly because the action is unlikely to be selected and the Q-values are only adapted slowly, especially when using a small learning rate. To address this issue, we investigated a variant of the FAQ-Learning technique, proposed by *Kaisers et al.* for multi-agent RL [26]. In the proposed technique, the Q-values are updated as defined by Equation (2.3) where $P(s,a)$ is the probability of taking action a in state s.

$$Q(s,a) = Q(s,a) + \min\left(\frac{\alpha}{P(s,a)}, 1\right) e(s,a)\left[r + \gamma \max_b Q(s',b) - Q(s,a)\right] \tag{2.3}$$

Using this update rule, updates are percolated faster when an action has low selection probability. The cut-off at 1 is needed to avoid overflow of Q-values.

2.4.4 Exploration policy

One of the most challenging tasks in RL can be found in balancing between exploration and exploitation [27]. An often used approach to this tradeoff is the ϵ-greedy method [28]. Using this method, exploration comes down to random action selection and is performed with probability ϵ. The best action with respect to current estimates is thus exploited with probability $1 - \epsilon$. Since the optimal configuration of the ϵ-parameter is very application dependent, rigorous tuning is required to obtain desirable results.

Another commonly used exploration method is Softmax [25]. In contrast to the ϵ-greedy method, with Softmax, action-selection is always performed in a probabilistic way. A Boltzmann distribution is used to rank the learned Q-values, based on which selection probabilities $P(s, a)$ are calculated using Equation (2.4) for every state-action pair (s, a). The positive parameter β is called the *inverse temperature*. As with the ϵ-greedy method, the parameter has to be tuned to balance the exploration rate for the specific application.

$$P(s, a) = \frac{e^{\beta Q(s,a)}}{\sum_b e^{\beta Q(s,b)}} \tag{2.4}$$

Tokic et al. propose the Value-Difference Based Exploration with Softmax action selection (VDBES) policy [27, 29]. With VDBES, the ϵ-greedy and the Softmax policy are combined in a way that exploration is performed, using the Softmax probabilities defined in Equation (2.4), with probability ϵ. Greedy action selection is executed with probability $1 - \epsilon$. Furthermore, a state-dependent exploration probability $\epsilon(s)$ is used instead of defining a global parameter. The $\epsilon(s)$ values are updated in every learning step, based on the difference in Q-values before and after that step, denoted as Δ. In this way, the agent is guided to be more explorative when knowledge about the environment is uncertain, indicated by large fluctuations in Q-values. When the agent's knowledge becomes certain however, the amount of exploration should be reduced. This behavior is achieved by updating the $\epsilon(s)$ values according to Equation (2.5).

$$\epsilon(s) = \delta \frac{1 - e^{\frac{-|\Delta|}{\sigma}}}{1 + e^{\frac{-|\Delta|}{\sigma}}} + (1 - \delta)\epsilon(s) \tag{2.5}$$

In this equation, the *inverse sensitivity* σ influences the exploration in a way that higher values of σ allow high levels of exploration only when the Q-value changes are large. Lower σ-values allow exploration even at smaller Q-value changes. The parameter $\delta \in [0; 1]$ defines the relative weight of the selected action on the state-dependent exploration probability. A commonly used value for δ is the inverse of the number of actions since all actions should contribute equally to $\epsilon(s)$.

The proposed HAS client has been evaluated using both the Softmax and the VDBES exploration policy.

Table 2.1: Proposed environmental state definition.

State element	Range	Levels
Buffer filling	[0 ; B_{max}]sec	$\frac{B_{max}}{T_{seg}} + 1$
Bandwidth	[0 ; BW_{max}]bps	$N + 1$

2.4.5 State & reward definition

In our initial approach to a self-learning HAS client [11], we proposed an environment model with six state variables, yielding over 2.5 million discrete states in the evaluated scenario. Using this large state definition, convergence issues arose, making the client inapplicable in situations with variable bandwidth. Based on this experience, the proposed learning agent uses a state definition, constructed of only two parameters: the current client buffer filling level and the available bandwidth perceived by the client. Both elements are continuous values and thus need to be discretized in order to apply the previously presented RL algorithms. The specific value ranges and number of discretization levels are shown in Table 2.1. In this table, B_{max} and T_{seg} respectively denote the maximum client buffer size and the segment duration in seconds. The number of quality levels and the highest possible throughput, e.g. the physical link capacity, are represented by N and BW_{max} respectively. In the considered scenario with a maximum client buffer size of 20s and a video stream with segments of 2s, available at 7 quality levels, this yields an environment model with 88 states.

Since the reward function is the fundamental guide for the RL agent to learn the desired policy, we want the reward function to be a measure for the QoE. For the construction of this reward function, three aspects of quality are considered, as identified by *Mok et al.* [30]: (i) the current video quality level, (ii) the switching in quality levels during the video playout and (iii) buffer starvations, leading to video freezes. The reward components for each of these aspects are constructed as shown in Equations (2.6), (2.7) and (2.8). For the quality level and switching components, a linear function is used. The buffer filling component has been modeled using a linear function for a non-empty buffer as well, but a large punishment of -100 is given when the buffer is empty to avoid video freezes.

$$\mathrm{R}_{\mathrm{quality}} = QL_i - N \tag{2.6}$$

$$\mathrm{R}_{\mathrm{switches}} = -1.0 * |QL_i - QL_{i-1}| \tag{2.7}$$

$$\mathrm{R}_{\mathrm{bufferfilling}} = \begin{cases} -100 & : B_i = 0 \\ B_i - B_{max} & : B_i > 0 \end{cases} \tag{2.8}$$

As shown in Equation (2.9), the total reward function is defined as the sum of these components.

$$\mathrm{R} = \mathrm{R}_{\mathrm{quality}} + \mathrm{R}_{\mathrm{switches}} + \mathrm{R}_{\mathrm{bufferfilling}} \tag{2.9}$$

2.4.6 Action definition

The agent's action is to select which quality level to download for every video segment. As described in Section 2.2, a HAS client selects the quality level for the next segment upon arrival of the previous segment. The set of available actions corresponds to the available quality levels of the video sequence and is therefore static throughout the playout of a video sequence. The concrete number of actions depends on the video sequence. The quality levels of the video sequence used in this work are described in Section 2.6.1.

2.5 Initial Q-value estimation

2.5.1 Rationale

When learning starts, the agent has no knowledge about the environment and the quality of the actions. Therefore, the Q-values $Q(s, a)$ are initialized at a default value, regularly $Q(s, a) = 0$. Since the reward function, described in Section 2.4.5, produces negative values, unexplored state-action combinations will always be favored by the exploration policy since they have the highest Q-values. Using strong negative default values would have the opposite effect, favoring previously used actions.

For the client to be able to react to new, unseen states in an acceptable way, domain knowledge can be incorporated into the initial Q-Tables. However, care has to be given to the magnitude of the Q-values in order not to fully restrict learning steps. When the initial Q-values magnitude is too high, the level of exploration is too limited, hampering the ability to find an acceptable solution. The goal is to design an initial Q-Table that allows to achieve higher performance when learning in unseen states, while reaching similar results as with standard Q-Learning in the converged state.

2.5.2 Estimation algorithm

Algorithm 2.1 estimates the average reward for every state-action pair, to be used as initial Q-values. The algorithm works as follows. First, for every state-action pair, the quality and buffer reward component are estimated. Based on the average bandwidth in the discrete bandwidth level bw and the average segment size for quality level ql, the estimated download duration $expectedDur$ is calculated. However, the actual bandwidth, available when downloading the next segment,

Output: Matrix $estimates$ with estimated Q-values

Initialize value estimate matrix $estimates$
for all Discrete buffer filling level buf **do**
  for all Discrete bandwidth level bw **do**
    ▷ Estimate quality and buffer reward
    for all Quality level ql **do**
      $expectedDur \leftarrow \frac{bitrate[ql]*segmentDur}{averageBW[bw]}$
      $changeProb \leftarrow \frac{expectedDur}{300}$
      $totReward \leftarrow 0.0$
      for all Discrete bandwidth level $nextBw$ **do**
        $transProb \leftarrow \begin{cases} 1.0 - changeProb & : bw = nextBw \\ \frac{changeProb}{numBWs-1} & : else \end{cases}$
        $duration \leftarrow \frac{bitrate[ql]*segmentDur}{averageBW[nextBw]}$
        if $duration < segmentDur$ **then**
          $expectedChange \leftarrow \lfloor \frac{segmentDur}{duration} \rfloor$
        else
          $expectedChange \leftarrow -1 * \lceil \frac{duration}{segmentDur} \rceil$
        end if
        $newBuf \leftarrow buf + expectedChange$
        $reward \leftarrow (ql - maxQl) + ((newBuf * segmentDur) - maxBuf)$
        $totReward \leftarrow totReward + (transProb * reward)$
      end for
      $estimates[buf][bw][ql] \leftarrow totReward$
    end for
    ▷ Estimate the average selected quality level
    $avgQl \leftarrow 0.0$
    for all Quality level ql **do**
      $prob \leftarrow$ probability of taking action ql in the specified state
      $avgQl \leftarrow avgQl + (prob * ql)$
    end for
    ▷ Estimate switch reward
    for all Quality level ql **do**
      $reward \leftarrow estimates[buf][bw][ql]$
      $switchReward \leftarrow -1 * |ql - avgQl|$
      $estimates[buf][bw][ql] \leftarrow reward + switchReward$
    end for
  end for
end for

Algorithm 2.1: Initial Q-value calculation algorithm.

could vary. Therefore, we calculate the probability $changeProb$ that the available bandwidth level will change in the next $expectedDur$ seconds. To calculate this probability, we make the assumption that the available bandwidth remains stable for a uniform distributed amount of time between 1s and 300s. With probability $1 - changeProb$, the bandwidth level will stay the same, while any other bandwidth level has an equal probability $\frac{changeProb}{numBWs-1}$ to occur. For every bandwidth level $nextBw$, the average download duration $duration$ is calculated. Based on this duration and the segment duration $segmentDur$, the expected buffer change and new buffer filling level $newBuf$ can be calculated. Using this information, the quality and buffer filling reward components when the actual bandwidth is $nextBw$ are estimated. The influence of this value on the total reward is weighted by the probability $transProb$ that the available bandwidth level will be $nextBw$.

The next step in the algorithm is to estimate the average switch reward for every state-action pair. Therefore, the average quality level, selected in the specified state is calculated. In this calculation, the probability of taking an action in a certain state is given by the Softmax action-selection probability, defined in Equation 2.4. Knowing the average selected quality level $avgQl$, the average switch depth when selecting quality level ql can be estimated as $|ql - avgQl|$. Using this value, the estimated switch reward and the resulting total reward is calculated.

It is important to note that this algorithm is based on some assumptions and approximations. For example, a model for the available bandwidth is assumed when calculating the bandwidth shifting probability and averages are used when estimating the download duration. Therefore, the resulting values are only initial estimates and the self-learning client is needed to adapt the behavior to the actual network environment. The complexity of the algorithm is linear in the number of discrete buffer filling levels and quality levels, and quadratic in the number of discrete bandwidth levels. Given the limited state-action space and the fact that the initial Q-values are only calculated once offline, the execution time is negligible.

2.6 Performance evaluation

2.6.1 Experimental setup

The experiments have been performed using the NS-3[6] based simulation framework described by *Bouten et al.* [31]. A network topology, shown in Figure 2.2, has been modeled, consisting of a single HAS client and server. This topology corresponds to a typical DSL access network scenario. On the last link on the path between the server and the client, a bandwidth capacity of 4Mbps is available for video delivery. At the client side, a maximum of 20s can be buffered.

[6] http://www.nsnam.org

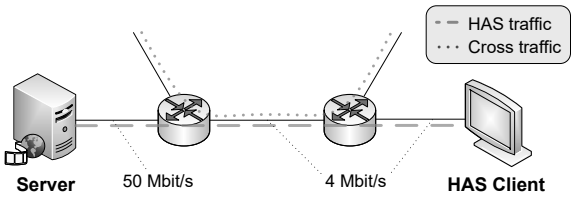


Figure 2.2: Overview of the simulated topology.

Table 2.2: Quality levels and corresponding bit rates.

Quality level	Bit rate
1	300kbps
2	427kbps
3	608kbps
4	866kbps
5	1233kbps
6	1636kbps
7	2436kbps

Since the goal of the self-learning HAS client is to be able to deal with variable network environments, a highly variable bandwidth trace has been constructed by simulating cross traffic over a 3Mbps link and measuring the available throughput at the client side. The generated cross traffic is a sequence of bandwidth bursts, normally distributed between 0kbps and 2640kbps with a granularity of 264kbps. Each burst persisted for a uniformly distributed amount of time ranging from 1s to 300s. Using the resulting bandwidth trace not only yields high variability within an episode, but also across the episodes.

On this topology, the *Big Buck Bunny* video trace was streamed. A single episode of the video trace consists of 299 segments, each with a fixed length of 2s. Each segment has been encoded at 7 different quality levels, with bit rates ranging from 300kbps to 2436kbps, as shown in Table 2.2. To ensure the learning agent has time to converge, 400 episodes of the video trace are simulated.

The traditional Microsoft ISS Smooth Streaming (MSS) algorithm[7] is used to compare the behavior of the learning client to current deterministic HAS algorithms. Using MSS, three buffer thresholds should be tuned to configure the client behavior. For the panic, lower and upper buffer thresholds, the values of 25%, 40% and 80%, empirically determined by *Famaey et al.* [32], have been used in our experiments.

[7]Original source code available from:
https://slextensions.svn.codeplex.com/svn/trunk/SLExtensions/AdaptiveStreaming

To be able to draw meaningful conclusions when comparing the performance of the different clients, paired t-tests have been performed. Using paired t-tests, the significance of the difference between two approaches, applied in the same environment, can be shown. Furthermore, the comparison graphs in Section 2.6.3 contain error bars, visualizing the standard deviation of the plotted averages.

2.6.2 Evaluation metrics

The reward function, described in Section 2.4.5, has been constructed to be a measure of the quality of a decision on a single segment quality level. To evaluate the different approaches however, a measure of the total video playout quality has to be used. The QoE can only be estimated, either by subjective evaluation by a test panel or using an objective model of the users' perception. QoE of HAS video is still an active research topic and only a limited number of objective metrics are available. *De Vriendt et al.* define the QoE of HAS video to be dependent on the average segment quality and the standard deviation of the segment quality [33]. The parameters of the proposed *quality level model* were tuned based on the results of a small subjective test.

Next to the average quality level and the switching behavior, video freezes are also considered to heavily impact the QoE of video delivery. However, video freezes are not considered in the model proposed by *De Vriendt et al.* [33]. The influence of video freezes depends both on the number and the average length of freezes [30]. The calculation, proposed by *Mok et al.*, uses only three discrete levels of freeze frequency and length [30]. Based on an interpolation of these levels, a continuous function has been constructed to measure the impact of video freezes on QoE. The resulting function is shown in Equation (2.10), where F_{freq} and FT_{avg} represent the freeze frequency and the average freeze time respectively. Given that this function evaluates to 0 when no freezing occurs, the Mean Opinion Score (MOS) calculation proposed by *De Vriendt et al.* remains valid in a scenario without freezes [33].

$$\phi = \frac{7}{8} * \max\left(\frac{\ln(F_{freq})}{6} + 1, 0\right) + \frac{1}{8} * \left(\frac{\min(FT_{avg}, 15)}{15}\right) \tag{2.10}$$

Combining the quality level, switching and video freezing aspects, the estimated MOS for the playout of a HAS video, consisting of K segments, playing quality level QL_k for segment k, can be calculated using Equation (2.11). In this equation, the average played quality level and its standard deviation are respectively represented by $\mu = \frac{\sum_{k=1}^{K} QL_k}{K}$ and $\sigma = \sqrt{\frac{\sum_{k=1}^{K} (QL_k - \mu)^2}{K}}$. One can verify that the theoretical range of this metric in a scenario with seven quality levels $[1; 7]$ is $[0.00; 5.84]$. During the simulations however, a practical metric range $[0.00; 5.06]$ was observed, which corresponds to the typical levels of a MOS.

Table 2.3: Overview of evaluated parameter configurations.

	Parameter	Evaluated values
α	Learning rate	0.1 , 0.3 , 0.5 , 0.7 , 0.9
γ	Discount factor	0.1 , 0.3 , 0.5 , 0.7 , 0.9
λ	Eligibility trace-decay	0.1 , 0.5 , 0.6 , 0.7 , 0.9
β	Softmax inverse temperature	0.1 , 0.5 , 1.0 , 5.0

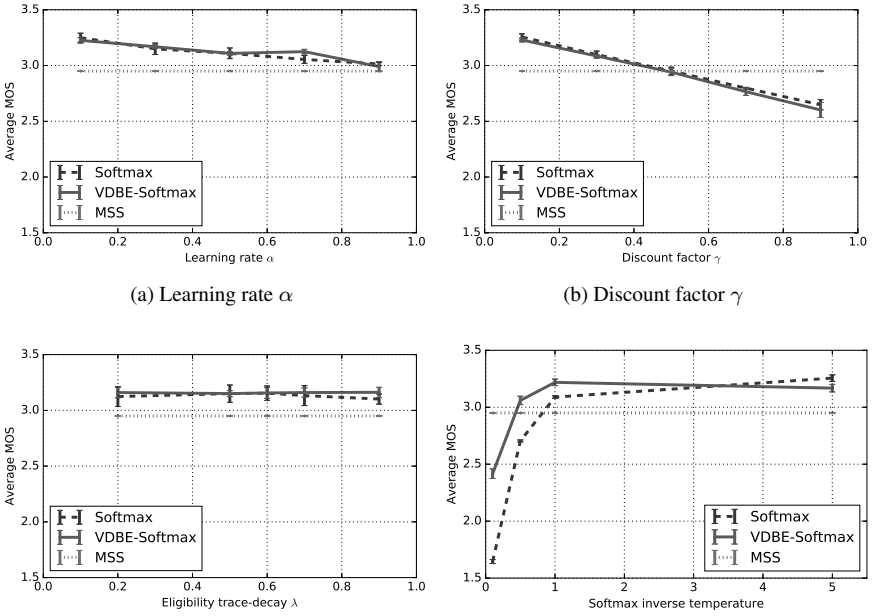
$$\text{MOS}_{est} = \max\left(0.81 * \mu - 0.96 * \sigma - 4.95 * \phi + 0.17, 0\right) \tag{2.11}$$

2.6.3 Results discussion

2.6.3.1 Parameter analysis

As described in Section 2.4, both the Q-Learning algorithm and the exploration policies contain multiple parameters that can be tuned to optimize the behavior. Given the continuous nature of the parameters and the mutual influence between them, it is unfeasible to evaluate all configurations to find the optimum. Therefore, a subset of 500 configurations has been evaluated for both the Softmax and the VDBES policy using the Q-Learning algorithm with default initial Q-values. Based on preliminary experiments, the VDBES inverse sensitivity parameter has been fixed to $\sigma = 1.0$. Using these configurations, the influence of every parameter can be analyzed and an acceptable configuration can be selected. An overview of the evaluated parameter values can be found in Table 2.3. For the learning rate and discount factor, an evenly spaced selection of the value range was taken. The evaluated values for the eligibility trace-decay were centered around 0.6, empirically found to be a good performing configuration. For the selection of Softmax inverse temperature values, preliminary experiments have shown that the influence of the parameter fades out for values above 1.0.

To consider the converged state, for every configuration, the MOS has been calculated over the last 50 of 400 episodes. For each parameter, the average MOS of the best 5 configurations for every evaluated value of that parameter has been calculated. The results are shown in Figure 2.3. It is clear that each of the parameters has similar influence for both exploration policies. While a clear trend is shown for the learning rate, discount factor and Softmax inverse temperature, the system is rather insensitive to the eligibility trace-decay value. This behavior can be explained by the strong preference of the system to low discount factors, strongly accelerating the decay, as defined by Equation (2.1). Since the VDBES approach only applies the Softmax formula when exploring, the VDBES approach is less sensitive to the value of the Softmax inverse temperature β. Finally, a wide range of parameters is shown to outperform the deterministic MSS algorithm.



(a) Learning rate α

(b) Discount factor γ

(c) Eligibility trace-decay λ

(d) Softmax inverse temperature β

Figure 2.3: Analysis of parameter influence.

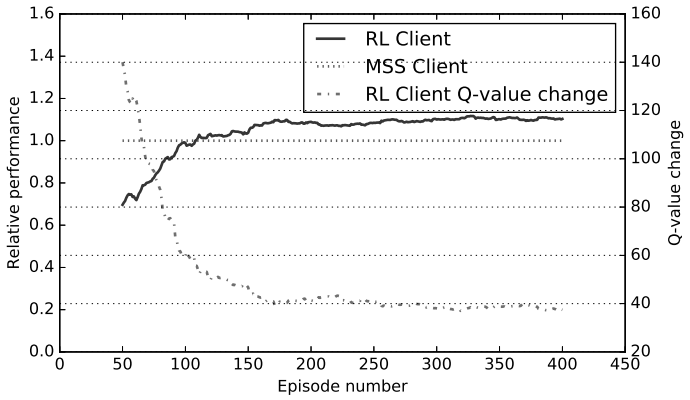


Figure 2.4: Convergence of the self-learning client performance, relative to the traditional MSS client.

Based on the analysis, the best configuration is determined to be $\alpha = 0.1$, $\gamma = 0.1$, $\lambda = 0.6$ and $\beta = 5.0$ using the Softmax exploration policy. Figure 2.4 shows the relative performance of the self-learning HAS client using this configuration compared to the traditional MSS client on the left axis. A moving average of the metric values of the last 50 episodes is presented in order to observe the general trend over the variable bandwidth episodes. The figure shows that after about 100 learning episodes, the client is able to achieve the same level of performance as the MSS client. The increasing trend stabilizes after about 200 episodes. The convergence of the learning agent can also be seen in the flattening out of the Q-value changes, plotted on the right axis. In the converged state, the self-learning HAS client is able to outperform the traditional MSS client with on average 10.31% in terms of average MOS in the last 50 episodes in a highly dynamic bandwidth environment. The performance increase is statistically significant with significance level 0.05 (two-tail paired t-test: $t = 8.5425$, $t_c = 2.0096$). With respect to the individual MOS factors, the MSS client is outperformed by 0.85%, 19.54% and 11.76% in terms of average quality level, average quality standard deviation and total freeze time respectively.

2.6.3.2 Frequency Adjusted Q-Learning

In Section 2.4.3, we argued that FAQ-Learning could possibly increase the performance of standard Q-Learning in strongly variable environments. To allow fair comparison between the performance of the FAQ-Learning and the standard Q-Learning client, both techniques have been applied in a highly dynamic bandwidth environment, using the parameter configuration selected in the previous section.

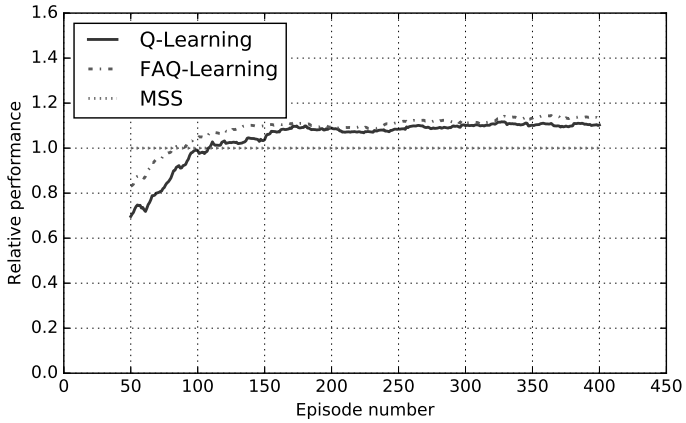


Figure 2.5: Relative performance of the FAQ-Learning and default Q-Learning approach compared to the traditional MSS client.

Table 2.4: Performance comparison of the MSS, Q-Learning and FAQ-Learning client in terms of MOS and freeze time.

Client	**Average MOS**	σ_{MOS}	**MOS Change***	**Total Freeze**	**Freeze Change***
MSS	2.94986	0.65923	–	13.975s	–
Q-Learning	3.25403	0.63482	+10.31%	12.332s	-11.75%
FAQ-Learning	3.35369	0.59948	+13.69%	4.667s	-66.60%

*Compared to the traditional MSS client.

In these simulations, again default initial Q-values have been used.

Figure 2.5 shows the relative performance of both clients compared to the traditional MSS client. The proposed FAQ-Learning approach clearly outperforms default Q-Learning, both in terms of convergence speed and absolute values. The performance increase is largely due to the smaller amount of freeze time. In Table 2.4, the performance of the FAQ-Learning client in the last 50 episodes is compared to the standard Q-Learning and MSS client in terms of average MOS and total freeze time. Using the proposed FAQ-Learning technique, the self-learning HAS client is able to outperform the traditional MSS client by 13.69% in terms of average MOS. This performance increase is statistically significant with significance level 0.05 (two-tail paired t-test: $t = 11.7688$, $t_c = 2.0096$). For the average quality level, average quality standard deviation and total freeze time, the achieved gain amounts 0.23%, 26.41% and 66.60% respectively.

Compared to the approach using standard Q-Learning, a statistically significant (two-tail paired t-test: $t = 3.6382$, $t_c = 2.0096$) average MOS increase of 3.06%

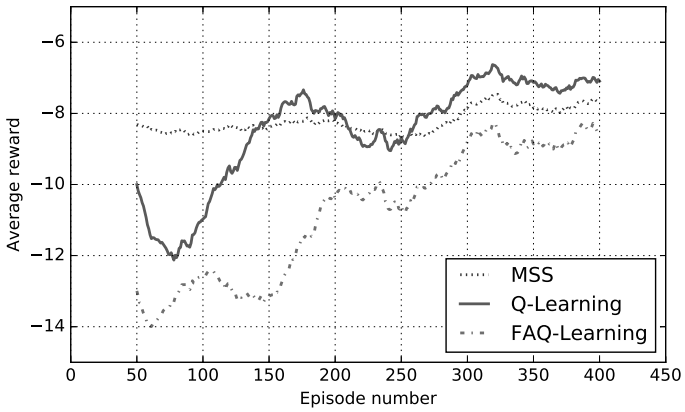
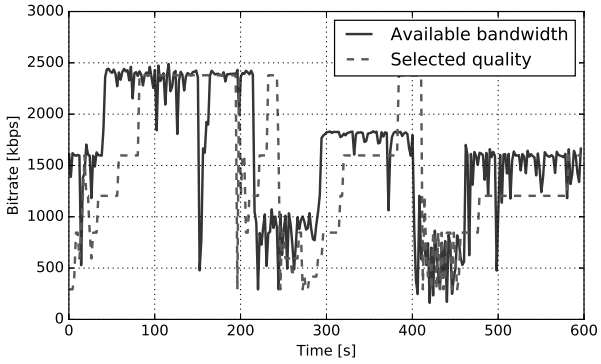


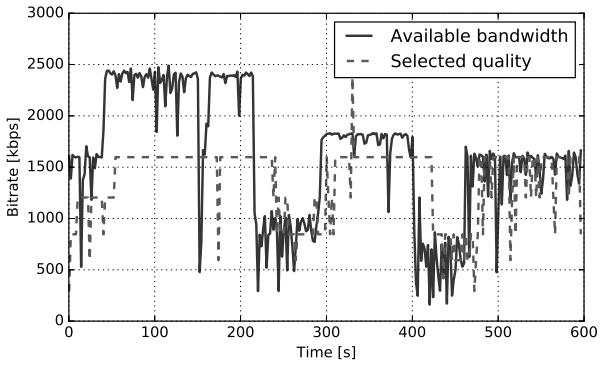
Figure 2.6: Reward performance of the FAQ-Learning and default Q-Learning approach compared to the traditional MSS client.

is obtained when applying FAQ-Learning. Despite of the performance gain in terms of QoE, the learning behavior in terms of reward values is inferior to the approach using standard Q-Learning. As shown in Figure 2.6, lower reward values are obtained when applying FAQ-Learning, compared to the default Q-Learning approach. The explanation for these conflicting results can be found by analyzing the resulting quality selection behavior of the three clients.

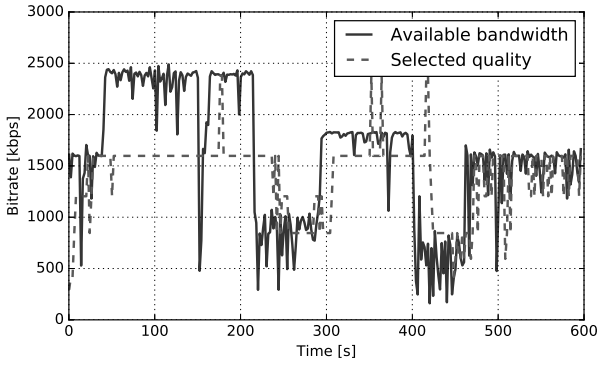
The quality selection behavior of the MSS, Q-Learning and FAQ-Learning clients in episode 375 is illustrated in Figure 2.7. The resulting average reward component values for this episode are shown in Table 2.5. Both Figure 2.7 and Table 2.5 show that even though the MSS client is able to reach higher quality levels at some points, the Q-Learning client achieves overall higher average quality, lower standard deviation of quality level and higher average buffer filling. The Q-Learning client thus results in more stable behavior. Moreover, it can be seen that using the FAQ-Learning client further increases the average quality level and decreases the quality level standard deviation. However, this comes at the cost of a lower buffer filling level, resulting in an overall lower reward value. Since buffer filling level is not directly influencing the QoE, the resulting MOS is not affected as long as the buffer is not fully depleted. As previously shown in Table 2.4, the lower buffer filling level does not introduce additional freezing time. Since the MOS is the aspect we aim to optimize in this use-case, the FAQ-Learning approach is preferred over the Q-Learning approach, despite of the lower average reward.



(a) Microsoft ISS Smooth Streaming



(b) Q-Learning



(c) FAQ-Learning

Figure 2.7: Behavior comparison of the MSS, Q-Learning and FAQ-Learning HAS clients in episode 375 of the variable bandwidth scenario.

Table 2.5: Reward components of the MSS, Q-Learning and FAQ-Learning clients in episode 375 of the variable bandwidth scenario.

Client	Reward component			
	Quality	**Switches**	**Buffer filling**	**Total**
MSS	-1.826	-0.365	-5.438	-7.629
Q-Learning	-1.632	-0.318	-4.294	-6.244
FAQ-Learning	-1.512	-0.278	-6.916	-7.950

2.6.3.3 Initial Q-value estimation

In Section 2.5, we proposed an algorithm to incorporate HAS domain knowledge into the initial Q-Table. Using this Q-Table, we target to drastically improve the client performance in the learning phase while maintaining a similar performance level as with default Q-Tables in the converged state. The use of pre-calculated Q-Tables has been evaluated on four bandwidth traces with different levels of variability, using the Q-Learning technique.

- **Fixed:** throughout the entire simulation, a fixed bandwidth level of 2Mbps is maintained.
- **Sinus:** the bandwidth level is modeled by a sine function with a period of 600s and a co-domain of [1Mbps;2Mbps].
- **Stepfunction:** every 20s, the bandwidth level switches between 1Mbps and 2Mbps.
- **Variable:** the highly variable bandwidth trace, as described in Section 2.6.1.

Figure 2.8 shows the performance of both approaches and the MSS client in the learning and converged state for each of the four bandwidth configurations. As in the rest of this chapter, we refer to the converged state as the last 50 episodes of the simulation of 400 episodes. The learning phase is defined as the first 50 episodes. In each of the bandwidth configurations, the self-learning client is shown to benefit from using the calculated Q-Tables in the learning phase. The statistical significance of the average MOS changes is presented in Table 2.6. The slight performance decrease in the converged state is caused by the reduced learning possibilities, introduced by the domain knowledge. However, the converged results are comparable to the performance when using default Q-Tables. Furthermore, the figure shows that the self-learning client, once converged, is able to outperform the deterministic MSS client on average by 11.18% (for the variable bandwidth configuration) to 18.89% (for the sinus bandwidth configuration) in terms of average MOS, depending on the bandwidth configuration. For the variable network configuration, the average quality level, average quality standard deviation and total freeze time have gained 1.52%, 20.38% and 5.45% respectively.

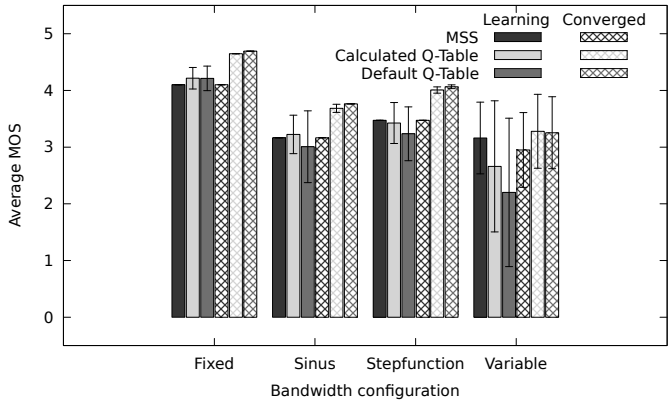


Figure 2.8: Performance comparison of the traditional MSS client and the self-learning client using default and calculated initial Q-Tables in the learning and converged phase.

Table 2.6: Statistical significance of average MOS differences using calculated initial Q-Tables. Significance results are obtained by two-tail paired t-testing with significance level 0.05.

Bandwidth Configuration	**Phase**	**Default Q-Table**	**Calculated Q-Table**	**T-test Result**	**Stat. Sign.***
Fixed	Learning[a]	4.21272	4.21582	0.1362	
	Conv.[b]	4.69312	4.64692	83.4822	✓
Sinus	Learning[a]	3.00777	3.2242	2.8279	✓
	Conv.[b]	3.76258	3.68460	7.6924	✓
Stepfunction	Learning[a]	3.23553	3.42564	2.8127	✓
	Conv.[b]	4.06532	4.00774	6.3059	✓
Variable	Learning[a]	2.20202	2.66074	4.7204	✓
	Conv.[b]	3.25403	3.27974	0.9528	

*Critical t-value: $t_c = 2.0096$.
[a]First 50 of 400 episodes.
[b]Last 50 of 400 episodes.

Table 2.7: Performance comparison of the Q-Learning-based client using default and calculated initial Q-Tables in terms of average MOS and total freeze time for the variable bandwidth configuration.

Phase	Initial Q-Table	Avg. MOS	σ_{MOS}	MOS Change*	Total Freeze	Freeze Change*
Learn.[a]	Default	2.20202	1.30955	–	526.075s	–
	Calc.	2.66074	1.15649	+20.83%	252.460s	-52.01%
Conv.[b]	Default	3.25403	0.63482	–	12.332s	–
	Calc.	3.27974	0.65199	+0.79%	13.214s	+7.15%

*Compared to the client using default Q-Tables.
[a]First 50 of 400 episodes.
[b]Last 50 of 400 episodes.

Besides from increasing the performance in terms of average MOS, compared to default initial Q-Tables, incorporating domain-knowledge in the initial Q-Tables strongly decreases the total freeze time in the learning phase. A comparison in terms of both MOS and total freeze time is given in Table 2.7 for the variable bandwidth configuration. The table shows that using pre-calculated initial Q-Tables compared to default values strongly boosts the client performance in the learning phase while reaching a similar performance level when converged. Even though an additional freeze time of about 900ms is introduced in the converged state, the overall MOS, incorporating the freezes, is not affected. The increased freeze time is compensated by higher average quality level and lower quality standard deviation.

2.6.3.4 Results summary

Table 2.8 summarizes the results of the self-learning client approaches, compared to the traditional Microsoft ISS Smooth Streaming (MSS) client, in terms of both the average MOS and the individual quality components. It is shown that a Q-Learning based HAS client outperforms the deterministic MSS client for each of the quality aspects. Using the proposed Frequency Adjusted Q-Learning (FAQ-Learning) technique, further improvement is obtained. Furthermore, the use of pre-calculated initial Q-Tables strongly boosts the performance in the learning phase and reaches similar results as with default Q-Tables in the converged state.

2.7 Conclusions

In this chapter, we designed a Reinforcement Learning (RL)-based HTTP Adaptive Streaming (HAS) client, dynamically adjusting its behavior to the perceived networking environment. We presented an extended parameter analysis to fine-

Table 2.8: Performance summary of the self-learning approaches in the variable bandwidth configuration, compared to the traditional MSS client, in terms of the quality components.

Technique	Initial Q-Table	MOS Change*	Quality Change*	Switching Change*	Freeze Change*
Q-Learning	Default	+10.31%	+0.85%	-19.54%	-11.76%
FAQ-Learning	Default	+13.69%	+0.23%	-26.41%	-66.60%
Q-Learning	Calc.	+11.18%	+1.52%	-20.38%	-5.45%

*Compared to the traditional MSS client.

tune the client configuration to operate in a dynamic network environment. Next, we proposed using a Frequency Adjusted Q-Learning (FAQ-Learning) approach to strongly increase the client performance in variable environments. Furthermore, we presented an estimation algorithm to incorporate domain knowledge into the initial Q-Tables. Using these estimations, we were able to drastically improve the clients performance during its learning phase, both in terms of average Mean Opinion Score (MOS) and total freeze time. The resulting self-learning HAS client is shown to outperform the deterministic traditional Microsoft ISS Smooth Streaming (MSS) client in terms of average MOS by 11% to 18% in all of the evaluated bandwidth scenarios with different degrees over variability, while increasing the performance for each of the identified MOS components.

Acknowledgment

M. Claeys is funded by grant of the Agency for Innovation by Science and Technology in Flanders (IWT). The research was performed partially within the ICON MISTRAL project (under grant agreement no. 10838). This work was partly funded by Flamingo, a Network of Excellence project (318488) supported by the European Commission under its Seventh Framework Programme. The Alcatel-Lucent research was performed partially within IWT project 110112.

References

[1] L. P. Kaelbling, M. L. Littman, and A. W. Moore. *Reinforcement learning: A survey*. Journal of Artificial Intelligence, 4(1):237–285, 1996.

[2] T. Stockhammer. *Dynamic adaptive streaming over HTTP: Standards and design principles*. In Proceedings of the ACM International Conference on Multimedia Systems, pages 133–144, 2011.

[3] S. Akhshabi, A. Begen, and C. Dovrolis. *An experimental evaluation of rate-adaptation algorithms in adaptive atreaming over HTTP*. In Proceedings of the ACM International Conference on Multimedia Systems, pages 157–168, 2011.

[4] C. Liu, I. Bouazizi, and M. Gabbouj. *Parallel adaptive HTTP media streaming*. In Proceedings of International Conference on Computer Communications and Networks (ICCCN), pages 1–6, 2011. doi:910.

[5] T. Lohmar, T. Einarsson, P. Frojdh, F. Gabin, and M. Kampmann. *Dynamic adaptive HTTP streaming of live content*. In Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, pages 1–8, 2011.

[6] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen. *Bitrate and video quality planning for mobile streaming scenarios using a GPS-based bandwidth lookup service*. In Proceedings of the IEEE International Conference on Multimedia and Expo (ICME), pages 1–6, 2011.

[7] V. Adzic, H. Kalva, and B. Furht. *Optimized adaptive HTTP streaming for mobile devices*. Applications of Digital Image Processing, 8135(1):1–10, 2011.

[8] R. Trestian, A.-N. Moldovan, O. Ormond, and G.-M. Muntean. *Energy consumption analysis of video streaming to Android mobile devices*. In Proceedings of the IEEE Network Operations and Management Symposium (NOMS), pages 444–452, 2012.

[9] D. Jarnikov and T. Özçelebi. *Client intelligence for adaptive streaming solutions*. In Proceedings of the IEEE International Conference on Multimedia and Expo (ICME), pages 378–389, 2011.

[10] S. Xiang, L. Cai, and J. Pan. *Adaptive scalable video streaming in wireless networks*. In Proceedings of the ACM International Conference on Multimedia Systems, pages 167–172, 2012.

[11] M. Claeys, S. Latré, J. Famaey, T. Wu, W. Van Leekwijck, and F. De Turck. *Design of a Q-learning-based client quality selection algorithm for HTTP adaptive video streaming*. In Proceedings of the Workshop on Adaptive and Learning Agents, pages 30–37, 2013.

[12] V. Menkovski and A. Liotta. *Intelligent control for adaptive video streaming*. In Proceedings of the IEEE International Conference on Consumer Electronics (ICCE), pages 127–128, 2013.

[13] Y. Fei, V. W. S. Wong, and V. C. M. Leung. *Efficient QoS provisioning for adaptive multimedia in mobile communication networks by reinforcement learning*. Mobile Networks and Applications, 11(1):101–110, 2005.

[14] V. Charvillat and R. Grigora. *Reinforcement learning for dynamic multimedia adaptation*. Journal of Network and Computer Applications (JNCA), 30(3):1034–1058, 2007.

[15] D. W. McClary, V. R. Syrotiuk, and V. Lecuire. *Adaptive audio streaming in mobile ad hoc networks using neural networks*. Ad Hoc Networks, 6(4):524–538, 2008.

[16] J. Cao. *Using reinforcement learning for agent-based network fault diagnosis system*. In Proceedings of the IEEE International Conference on Information and Automation (ICIA), pages 750–754, 2011.

[17] R. Bagnasco and J. Serrat. *Multi-agent reinforcement learning in network management*. In Proceedings of the International Conference on Autonomous Infrastructure, Management and Security (AIMS), pages 199–202, 2009.

[18] D. Vengerov. *A reinforcement learning approach to dynamic resource allocation*. Technical report, Sun Microsystems, 2005.

[19] G. Tesauro, N. K. Jong, R. Das, and M. N. Bennani. *On the use of hybrid reinforcement learning for autonomic resource allocation*. Cluster Computing, 10(1):287–299, 2007.

[20] M. Lee, D. Marconett, X. Ye, and S. J. B. Yoo. *Cognitive network management with reinforcement learning for wireless mesh networks*. IP Operations and Management, 4786(1):168–179, 2007.

[21] D. Niyato and E. Hossain. *A radio resource management framework for IEEE 802.16-based OFDM/TDD wireless mesh networks*. In Proceedings of the IEEE International Conference on Communications (ICC), pages 3911–3916, 2006.

[22] N. Ouferhat and A. Mellouk. *A QoS scheduler packets for wireless sensor networks*. In Proceedings of the IEEE International Conference on Computer Systems and Applications, pages 211–216, 2007.

[23] S. Parakh and A. Jagannatham. *Game theory based dynamic bit-rate adaptation for H.264 scalable video transmission in 4G wireless systems*. In Proceedings of the International Conference on Signal Processing and Communications, pages 1–5, 2012.

[24] N. Mastronarde and M. Van Der Schaar. *Fast reinforcement learning for energy-efficient wireless communication*. IEEE Transactions on Signal Processing, 59(12):6262–6266, 2011.

[25] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. Reinforcement Learning: An Introduction, 1998.

[26] M. Kaisers and K. Tuyls. *Frequency adjusted multi-agent Q-learning*. In Proceedings of the International Conference on Autonomous Agents and Multi-agent Systems, pages 309–316, 2010.

[27] M. Tokic and G. Palm. *Value-difference based exploration: Adaptive control between epsilon-greedy and softmax*. In Proceedings of the Ger, pages 335–346, 2011.

[28] C. Watkins. *Learning from delayed rewards*. PhD thesis, University of Cambridge, England, 1989.

[29] M. Tokic. *Adaptive ε-greedy exploration in reinforcement learning based on value differences*. In Proceedings of the German Conference on Advances in Artificial Intelligence, pages 203–210, 2010.

[30] R. Mok, E. Chan, and R. Chang. *Measuring the quality of experience of HTTP video streaming*. In Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM), pages 485–492, 2011.

[31] N. Bouten, J. Famaey, S. Latré, R. Huysegems, B. De Vleeschauwer, W. Van Leekwijck, and F. De Turck. *QoE optimization through in-network quality adaptation for HTTP adaptive streaming*. In Proceedings of the International Conference on Network and Service Management (CNSM), pages 336–342, 2012.

[32] J. Famaey, S. Latré, N. Bouten, W. Van de Meerssche, B. De Vleeschauwer, W. Van Leekwijck, and F. De Turck. *On the merits of SVC-based HTTP adaptive streaming*. In Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM), pages 419–426, 2013.

[33] J. De Vriendt, D. De Vleeschauwer, and D. Robinson. *Model for estimating QoE of video delivered using HTTP adaptive streaming*. In Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM), pages 1288–1293, 2013.

3

Hybrid Multi-tenant Cache Management for Virtualized ISP Networks

M. Claeys, D. Tuncer, J. Famaey, M. Charalambides, S. Latré, G. Pavlou and F. De Turck.

Published in Journal of Network and Computer Applications, June 2016.

⋆ ⋆ ⋆

While the previous chapter focused on client-side optimization of video delivery, this chapter proposes in-network optimizations in the scenario where Internet Service Providers (ISPs) deploy telco-Content Delivery Networks (telco-CDNs) to reduce the pressure on their network resources. An important trend is virtualization of storage and networking resources, which can open up new business models by enabling the ISP to simultaneously lease its telco-CDN infrastructure to multiple third parties. Previous work has shown that multi-tenant proactive resource allocation and content placement can significantly reduce the load on the ISP network. However, the performance of this approach strongly depends on the prediction accuracy for future content requests. In this chapter, a hybrid cache management approach is proposed where proactive content placement and traditional reactive caching strategies are combined. In this way, content placement and server selection can be optimized across tenants and users, based on predicted content popularity and the geographical distribution of requests, while simultane-

ously providing reactivity to unexpected changes in the request pattern. Based on a Video-on-Demand (VoD) production request trace, it is shown that the total hit ratio can be increased by 43% while using 5% less bandwidth compared to the traditional Least Recently Used (LRU) caching strategy. Furthermore, the proposed approach requires 39% less migration overhead compared to the proactive placement approach we previously proposed in [1] and achieves a hit ratio increase of 19% and bandwidth usage reduction of 7% in the evaluated VoD scenarios and topology.

3.1 Introduction

Over the last decade, video streaming services have become the principal consumers of Internet traffic. In 2014, video over Internet Protocol (IP) has been reported to account for 67% of all IP traffic while this share is predicted to grow to 80% by 2018 [2]. Furthermore, with the advent of 4K resolution and 3D video, the quality requirements for these services are becoming more stringent. Today, the prevalent method to deliver the video to the end-users relies on Content Delivery Networks (CDNs). In order to meet the growing quality requirements, CDNs aim at bringing the content closer to the clients to reduce both the latency and the bandwidth consumption. Currently, a common way for traditional CDNs, such as Akamai or Netflix, to bring their content to the edge of the network is to physically place part of their distributed storage infrastructure inside the Internet Service Provider (ISP) network or to connect it to a nearby Internet exchange point through manually-negotiated contractual agreements.

However, given that large CDNs control both the placement of the content and the server selection strategy (i.e., select from which location to satisfy each request) across their geographically dispersed storage infrastructure with only limited knowledge about the underlying network, they can put an immense strain on the resources of ISP networks [3]. This has resulted in increasing operational costs and decreasing revenues for the ISPs. Therefore, they have started to explore alternative business models and service offerings, leading to the deployment of Telco CDNs. As ISPs have global knowledge about the utilization of their network resources, controlling the storage of content deep inside their network allows them to reduce the bandwidth demand on their backbone infrastructure while significantly improving the service quality for the end-users.

Moreover, the advent of cloud computing, Software-Defined Networking (SDN) and Network Function Virtualization (NFV) technologies have enabled ISPs to virtualize their telco-Content Delivery Network (telco-CDN) infrastructures. This allows them to dynamically offer virtual storage and content delivery services at the edge of the network, redeeming traditional CDN providers from installing additional hardware. In previous work, we proposed a proactive cache management

system for ISP-operated multi-tenant telco-CDNs to optimize the content placement and server selection across tenants and users [1, 4]. The tenants specify the amount of storage capacity they want to lease, while the management framework decides on the allocation of the leased capacity across the storage infrastructure, the content placement and the server selection. Such a scenario allows the tenants to bring their content closer to the end-user without having to physically deploy dedicated storage infrastructure inside the ISP network. This reduces the installation and operational costs for the content provider, while the ISP has better control over its network resources. All of these decisions are based on the prediction of content popularity and the geographical distribution of requests, making the performance strongly dependent on the accuracy of the request prediction.

Proactively placing content inside the ISP network has multiple advantages over reactive cache replacement, for example reduced bandwidth consumption by performing content migrations during off-peak hours and delivering cache hits on the first request of popular content. However, given the dynamic nature of content popularity in a Video-on-Demand (VoD) scenario, the performance of the proactive cache management presented in previous work can significantly fluctuate over time. To deal with the uncertainties in the future request pattern, this chapter proposes a hybrid cache management system, which combines the benefits of both proactive and reactive content placement strategies. In the proposed approach, the total caching space available in the network is divided in such a way that part of it is reserved to proactively push content in the different caching locations, while the rest is used to implement reactive caching. As such, content placement and server selection can be optimized across tenants and users, based on the predicted content popularity and the geographical distribution of requests, while simultaneously providing reactivity to unexpected changes in the request pattern. Furthermore, frequently migrating content to reconfigure the proactive placement can significantly influence the total bandwidth usage. Therefore, in this chapter, the migration overhead is taken into account in the placement decisions.

The main contributions of this work are as follows. We redefine the Integer Linear Program (ILP) model of the multi-tenant content placement and server selection problem proposed in previous work [1, 4] to take into account the overhead introduced by the frequent content migrations and to reduce the level of detail required for the request predictions. In contrast to our previous work, which required predictions of accurate request timestamps, the updated ILP model only requires predictions of request aggregates. We also investigate the benefits that can be obtained in terms of performance improvement by augmenting the results of the content popularity prediction algorithm with exogenous information about the future request pattern, e.g., based on trends in social media [5]. In addition, we show how the caching space can be divided between proactive and reactive placement and investigate the added value when considering adaptivity in the proposed

hybrid caching approach. We evaluate the performance in terms of both network and caching metrics based on a real VoD request trace from a major European ISP. The results show that the proposed hybrid cache management approach can outperform both a purely proactive and a purely reactive approach while significantly reducing the migration overhead compared to our previous work.

The remainder of this chapter is structured as follows. Section 3.2 highlights related work on cache management in distributed storage infrastructure. Section 3.3 presents the proposed management architecture and describes the scenario under study. Next, the problem is formally modeled as an ILP in Section 3.4 and the hybrid cache division strategy is described. Section 3.5 introduces the setup used to evaluate the proposed approach while the evaluation results are detailed in Section 3.6. Finally, the main conclusions are presented in Section 3.7.

3.2 Related work

Hybrid cache management approaches that partition the storage space available at each caching location in order to implement different caching strategies were proposed by *Applegate et al.* [6] and *Sharma et al.* [7]. However, the effect of partitioning was investigated on few performance metrics only (network utilization and latency) based on arbitrary fixed ratios. Furthermore, in these approaches, the partitioning is performed on each cache individually. In contrast, this chapter proposes a network-level partitioning and provides a more systematic analysis as it relies on a wide range of ratios and highlights the effect based on both network and caching metrics. In addition, the benefits of updating the partitioning ratio in an adaptive fashion are also quantitatively evaluated.

In the last few years, there have been significant research efforts towards the development of proactive cache management approaches. While some of the proposals have focused on content placement strategy only [8–13], others have proposed new mechanisms to manage the redirection of user requests [14, 15]. Optimal solution structures for the combined problem of content placement and server selection have also been developed [6, 16, 17]. However, all of these consider a single provider scenario only, which is a subset of the problem we investigate.

One of the most relevant approaches for the problem investigated here is the work of *Laoutaris et al.* [18, 19], in which algorithms for the joint optimization of capacity allocation and object placement decisions under known topological and user demand information were developed. However, despite the similarity in terms of problem objectives, the proposed models cannot apply to our scenario as they disregard per node capacity constraints and assume hierarchical caching infrastructures only.

In parallel to these management strategies, research efforts have also focused on the development of new models and frameworks to support the interaction be-

tween ISPs and CDNs. These range from ISP-centric caching approaches [20, 21], which exclude CDNs from the delivery chain, to collaborative solutions [3, 15, 22], defining new models of cooperation between ISPs and CDNs. Another relevant initiative concerns the Content Delivery Network Interconnection (CDNI) working group of the Internet Engineering Task Force (IETF) which focuses on standardizing the communications between CDNs to allow interoperability between different vendors[1]. All of these approaches are targeting to improve content delivery performance. Intermediate solutions, such as the one proposed by co-authors of this chapter [23], have also been considered and rely on providing a limited capacity CDN service within ISP networks by deploying caches at network edges.

Finally, while this chapter focuses on a VoD service, proactive content placement has recently been investigated in the context of live video streaming [24], which focuses on the end-to-end optimization of the stream delivery. In contrast to our approach, this targets very short re-configuration intervals (in the order of milliseconds) and involves changing different parameters (e.g., video stream placement and bit rate encoding selection), which are features specific to this type of service.

To the best of the authors' knowledge, this chapter is the first to propose a hybrid cache management approach with a network-level cache division and support for multi-tenant scenarios.

3.3 Experiment description

In this section, the considered experiment is described. Section 3.3.1 introduces the ISP-based CDN service where capacity is leased to one or multiple content providers. In Section 3.3.2, the characteristics of the investigated VoD use-case are presented, based on which the request prediction strategy is introduced in Section 3.3.3. Finally, Section 3.3.4 discusses the limitations of popularity prediction algorithms in the considered use-case.

3.3.1 Caching scenario

In this chapter, a scenario is considered where a large-scale ISP operates a limited capacity CDN service by deploying caching points within its network, as depicted in Figure 3.1. However, it is important to note that the applicability of the proposed approach is not limited to this scenario. For example, the standardization effort in the CDNI working group of the IETF allows the proposed approach to be used on a larger scale and to communicate between different CDN vendors.

In the considered scenario, the set of network nodes consists of edge nodes, which represent access networks connecting multiple users in the same region,

[1] https://datatracker.ietf.org/wg/cdni

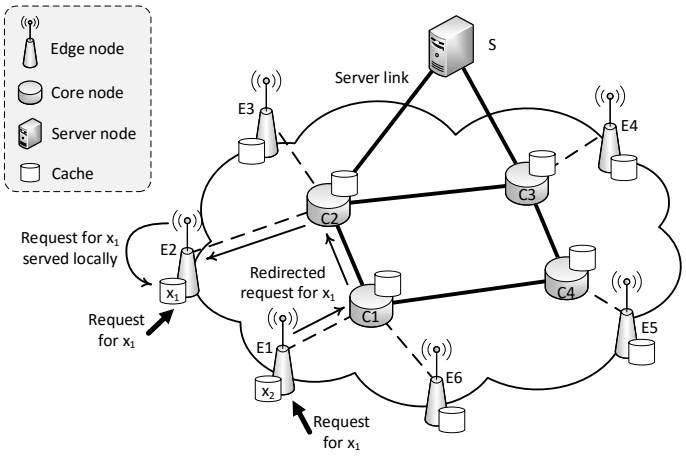


Figure 3.1: Overview of the telco-CDN service operated by the ISP.

and core nodes, which interconnect the different access networks. Each network node is equipped with caching capabilities, enabling a set of content items to be stored locally. These local caches could be external storage modules attached to routers or, with the advent of flash drive technology, integrated within routers.

All content requests are received at the edge nodes. If a requested content item is available in the local cache of the corresponding edge node, it is served locally. Otherwise, the request is redirected to one of the caches in the network where the requested item is stored. In case a copy of the item is not available in the ISP network, the request is served from the origin server outside of the network, hosted by the content provider. As depicted in Figure 3.1, the request for content x_1 received at edge node $E2$ is served locally, whereas the request for content x_1 received at edge node $E1$ is redirected to and served by node $E2$. In line with previous related research by *Applegate et al.* [6], we assume that traffic is routed over the shortest path, which was shown to be more realistic than arbitrary routing [25].

In this scenario, the ISP leases the caching space in its network to one or more content providers. Each content provider specifies the amount of caching capacity it wishes to lease for storing part of its content catalog, while the ISP decides which content items will be stored and where. The optimal placement of content in terms of network resource utilization depends on the geographical distribution of requests, the content popularity and the network topology. To minimize the resource utilization while simultaneously maximizing the total hit ratio (i.e., reducing the number of requests that have to be served from outside the network), this chapter proposes a hybrid multi-tenant cache management approach where the ISP controls the partitioning of the available storage space between the content

providers. In the proposed hybrid approach, part of the available caching capacity is allocated and managed according to a proactive cache management strategy, while the rest is controlled through a reactive approach. The proactive placement algorithm is executed periodically by a central manager to allocate the proactive part of the capacity across the network based on the predicted value of the content popularity and its geographical distribution. In contrast, the reactive approach is applied at each cache independently and serves as a buffer to react locally to unpredicted popularity changes.

3.3.2 VoD trace characteristics

To evaluate the proposed approach, a request trace of the VoD service of a leading European telecom operator has been used. The trace was collected between Saturday February 6, 2010 and Sunday March 7, 2010 and contains monitoring information for a period of 30 days. Due to a failure of the probing nodes, a couple of hours of monitoring data was missing for both February 12, 2010 and February 19, 2010. These gaps have been filled by considering the request pattern of the same period in the previous week, mapped on the content popularity of the same period in the last day. The resulting trace consists of 108,392 requests for 5644 unique videos, originating from 8825 unique users spread across 12 cities. Figure 3.2 shows the popularity curve of the VoD trace. In this chapter, all movies are considered to have an equal duration of 90 minutes and a bit rate of 1Mbit/s, resulting in a size of 675Mbyte for each video. The entire video catalog size thus amounts to about 3.64Tbyte. Each movie is requested in a segmented way, as is often the case in modern streaming technologies (e.g., Apple HLS, MPEG DASH), with a fixed duration of 1 second each. When multi-tenancy is considered, the set of movies in the VoD trace is uniformly split between the different content providers.

The daily number of requests and unique requested videos in the considered VoD trace are depicted in Figure 3.3. In this graph, a clear weekly pattern can be observed. The five peaks in the request pattern correspond to the five weekends, with increased activity on Friday, Saturday and Sunday. As only per-day aggregates are shown for the sake of visibility, the underlying diurnal trend cannot be observed. For Wednesdays and Sundays, the activity peak is situated between 4:30pm and 6:30pm, while for the other days of the week, the largest number of requests is reported between 8pm and 10pm. On average, 3613 requests for 1012 unique movies, i.e., about 18% of the total movie catalog, are monitored per day.

3.3.3 Request prediction

To be able to take proactive decisions concerning content placement, predictions about the content popularity in the considered period have to be made. Content popularity prediction is a complex issue that is outside the scope of this chapter.

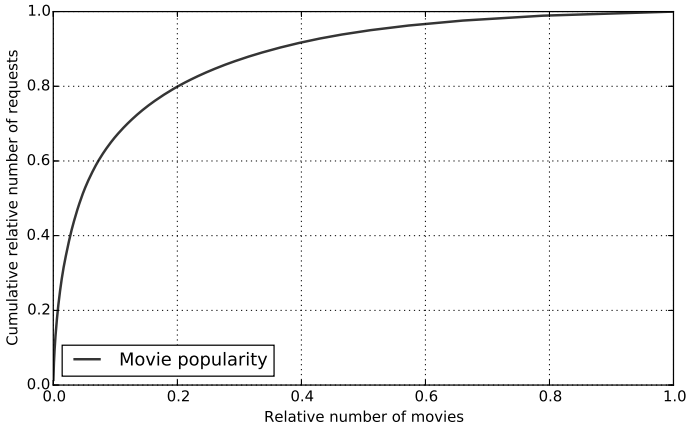


Figure 3.2: Popularity curve of the considered VoD trace.

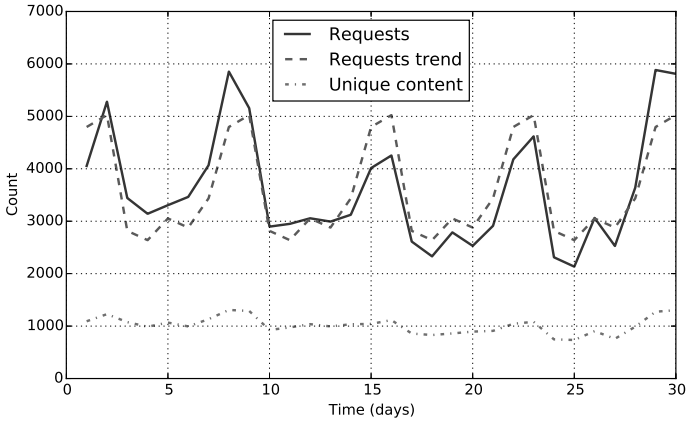


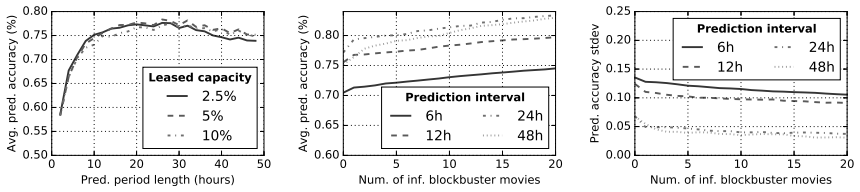
Figure 3.3: Request pattern and number of daily unique content items in the considered VoD trace.

Therefore, in this chapter we apply a simple request prediction strategy based on the characteristics of the VoD trace described in the previous section.

The request pattern consists of multiple types of information: (i) the request intensity, i.e., the total number of requests in the network, over time, (ii) the geographical distribution of the requests, and (iii) the relative content popularity, i.e., the distribution of the requests amongst all the content items. As shown in Figure 3.3, a clear weekly pattern can be identified for the request intensity of the VoD trace. Based on this observation, the request intensity and the geographical distribution of requests of the same period of the previous week are used for predicting the request pattern in a specific period. However, given the highly dynamic nature of video popularity, such an approach cannot be used to predict content popularity. Although some request intensity trends can be identified for each day of the week, it is more likely that the change in popularity of a given content item will be more significant over a week than between two consecutive days. Based on the results reported in our previous work [1], we use the content popularity of the last 3 days to predict the popularity of specific content items as it was shown to result in the highest prediction accuracy.

To estimate the quality of the prediction, we define an accuracy metric as follows. When the total amount of leased capacity is equal to the combined size of y videos, the accuracy of the y most popular content items in the request prediction is crucial for the performance of the system, as these items are most likely to be cached. Therefore, we define the prediction accuracy in a specific period as the ratio between the relative number of requests for the predicted y most popular videos, denoted as r_{pred}, and for the actual y most popular videos, denoted as r_{act}, over that period. For example, we consider a total amount of leased capacity of y =100 videos. If the set of 100 videos that are predicted to be the most popular accounts for r_{pred}=50% of all requests in the considered period while the actual 100 most popular videos during that period amount for r_{act}=65% of the requests, the accuracy of the popularity prediction is said to be 76.92% ($= \frac{r_{pred}}{r_{act}} = \frac{50\%}{65\%}$).

Figure 3.4a shows the average prediction accuracy for different amounts of leased capacity (expressed relatively to the total catalog size), in terms of the length of the predicted period. It can be observed that the prediction accuracy increases with the predicted period length, up to a length of 24h. This can be explained by the characteristics of the content popularity. Typically, requests for popular content are spread over the day while less popular content is only requested a few times a day at specific points in time. As the considered period of time decreases, the number of unique requested videos decreases and the steepness of the popularity curve significantly reduces. Therefore, the shorter the time period, the harder the content popularity prediction. In addition, due to the highly dynamic nature of the content popularity and the relatively short popularity lifetime of video content, the prediction accuracy degrades when the predicted period length exceeds 24h.



(a) Influence of the predicted period length on the average prediction accuracy with a history length of 3 days for different amounts of leased capacity.

(b) Influence of the knowledge about blockbuster movies on the average prediction accuracy with a history length of 3 days and a leased capacity of 5%.

(c) Influence of the knowledge about blockbuster movies on the standard deviation of the prediction accuracy with a history length of 3 days and a leased capacity of 5%.

Figure 3.4: Analysis of the prediction accuracy for the VoD trace.

Furthermore, it can be seen that the prediction accuracy is rather insensitive to the amount of leased capacity. However, this does not mean that the popularity of less popular content is as easy to predict as for more popular content. Because of the steep popularity curve of the considered VoD trace, shown in Figure 3.2, the vast majority of cache hits is for the most popular content, making these content items the decisive factor in the accuracy metric (which is based on cache hits). Therefore, the decreasing accuracy of predicting less popular content does not have a significant influence on the total prediction accuracy.

In the considered VoD request trace, a significant part of daily requests (17.92% on average) is for videos that were not requested in the recent history of 3 days. Even when longer history windows of up to one week are used, about 15% of the requests remains for unseen content. None of these requests can be predicted using pure history-based prediction techniques. The prediction of requests for new content is an active research area and previous research efforts have proposed to extract information about future blockbuster movies, for example based on activity trends on social media [5], using collaborative filtering models [26] or relying on life span patterns of video popularity [27, 28]. In this work, we assume that we are informed about a short list of the most popular blockbuster movies on a daily basis, regardless of the technique used to extract this information (i.e., a limited number of the most popular movies are assumed to be known by the system in advance, based on external information). Figure 3.4b and 3.4c show the influence of the number of informed blockbuster movies on the average prediction accuracy and its standard deviation, respectively. It can be seen that, for example in a scenario where predictions are made every 24h, even with a limited amount of 10 informed blockbuster movies (about 1% of the requested movies per day on average), the average prediction accuracy is increased by 5% while its standard deviation is decreased by 41% compared to the pure history-based prediction. This shows that

besides increasing the average prediction accuracy, using knowledge about blockbuster movies significantly stabilizes the prediction accuracy over time. For example, information about 10 blockbuster movies lowers the difference between the highest and the lowest prediction accuracy over time from 28% to 18%.

3.3.4 Popularity prediction limitations

As described in the previous section, exogenous information can be used to predict blockbuster movies. However, this can only be done for a small fraction of popular content. A significant part of new content remains unpredicted. In addition, Figure 3.5 shows the average shifting probabilities in the VoD trace. The plot shows the percentage of content items that shifts from a given popularity category to another category on average. It is important to note that while the different popularity categories are unevenly sized in terms of number of contents, each category accounts for a similar number of requests. For example, the 2.5% most popular content items on a given day account for 29% of the requests on average, while the 80% least popular content items (i.e., category 100% in Figure 3.5) account for 33% of the requests on average. It can be observed that the content popularity in the considered trace is very volatile. For example, on average, 24.60% of the 2.5% to 5% most popular movies on a given day are not requested at all the next day. Furthermore, it can be deduced from Figure 3.5 that on average more than 13% of the top 2.5% most popular content was not requested the day before[2]. These findings strongly limit the possibilities of popularity prediction techniques. When considering time periods shorter than 24h, the fluctuation in content popularity becomes even more significant, resulting in a lower prediction accuracy. Additionally, it can be observed from Figure 3.5 that the fluctuation in popularity grows for less popular content. Therefore, when less popular content is cached, it is more likely to be replaced in the next reconfiguration phase, resulting in additional migration overhead.

3.4 Hybrid cache management

As shown in previous work, proactive content placement can result in more efficient resource usage, compared to reactive cache management [1]. However, it was shown that the achievable performance gain strongly depends on the accuracy of the popularity prediction. As described in Section 3.3.4, the characteristics of the VoD trace limit the accuracy of the request prediction. To deal with these limitations, a hybrid caching approach is proposed, applying both proactive and

[2]On average, 82.07% of all content is not requested on a single day, 0.40% of which is in the 2.5% most popular content the next day. This amounts to 0.33% of the total content catalog, or 13.2% of the 2.5% most popular content.

Figure 3.5: Average popularity shifting probabilities in the considered VoD trace (*NR: Not Requested).

reactive caching techniques. A proactive content placement is performed periodically, based on a prediction of the content popularity, while part of the capacity is used for reactive caching in order to react to unexpected popularity fluctuations. It is important to note that even though the performance of both the proactive approach and the hybrid approach will be influenced by the quality of the request prediction, the hybrid approach will always benefit from the reactive caching part to deal with new popular content that will inevitably be missed by the request prediction strategy, as discussed in Section 3.3.4. Therefore, even though the optimal cache division might change, the hybrid approach will always outperform both the proactive and reactive caching approaches, or at least perform equally as good as the best of the two approaches.

First, some general notations are introduced in Section 3.4.1. Next, the hybrid cache division strategy is presented in Section 3.4.2. Finally, Section 3.4.3 describes the proactive placement algorithm.

3.4.1 General notations

To support the description of the algorithm, we first introduce some general notations used to represent the characteristics of the considered scenario. The notations are summarized in Table 3.1. The network topology is modeled as a directed graph $G = (N, L)$ with N and L representing the set of nodes and links, respectively.

Table 3.1: Summary of the general notations.

Notation	Description
N	Set of network nodes.
$S \in N$	Server node.
$N_{ISP} \subset N$	Set of ISP-managed nodes.
$N_C \subset N_{ISP}$	Set of core nodes.
$N_E \subset N_{ISP}$	Set of edge nodes.
L	Set of network links.
$L_S \subset L$	Set of ingress links.
$L_{ISP} \subset L$	Set of ISP-managed links.
$c_n \in \mathbb{N}^+$	Storage capacity of node $n \in N$.
$I_n \subseteq L$	Set of incoming links of node $n \in N$.
$O_n \subseteq L$	Set of outgoing links of node $n \in N$.
$c_l \in \mathbb{N}^+$	Bandwidth capacity of link $l \in L$.
$R_{s,d} \subseteq L$	Routing path from $s \in N$ to $d \in N$.
P	Set of content providers.
$d_p \in \mathbb{N}^+$	Caching space leased by $p \in P$.
O^p	Content items offered by $p \in P$.
$O = \bigcup_{p \in P} O^p$	Total set of offered content items.
$s_o \in \mathbb{N}^+$	Size of content item $o \in O$.
$b_o \in \mathbb{N}^+$	Bit rate of content item $o \in O$.

The set of nodes contains both the nodes N_{ISP}, belonging to the ISP network, and an external server node S, logically representing the Internet, containing all content of all providers. N_{ISP} can further be divided in a set of core nodes N_C and edge nodes N_E. The links L can be divided into a set of links, L_S, connected to the external server (i.e., the ingress links), and ISP-managed links L_{ISP}, connecting core and edge nodes. For each node $n \in N$, we define a caching capacity $c_n \in \mathbb{N}^+$ and a set of incoming and outgoing links, denoted by $I_n \subseteq L$ and $O_n \subseteq L$, respectively. For every link $l \in L$, the available bandwidth capacity is denoted by $c_l \in \mathbb{N}^+$. The routing strategy applied in the network is represented by a forwarding path $R_{s,d} \subseteq L$, for every source-destination pair $(s, d) \in N \times N$. The forwarding path can be divided into a set of server links $R^S_{s,d} \subseteq L_S$, containing the links in the forwarding path connected to the external server node S, and a set $R^{ISP}_{s,d} \subseteq L_{ISP}$ containing the other links in the forwarding path, inside the ISP network.

A set of content providers P lease caching space from the ISP. For each content provider $p \in P$, the leased amount of caching space and the set of offered content items are denoted by $d_p \in \mathbb{N}^+$ and O^p, respectively. $O = \bigcup_{p \in P} O^p$ represents the entire set of offered content. Every content item $o \in O$ has an associated size $s_o \in \mathbb{N}^+$ and bit rate $b_o \in \mathbb{N}^+$.

3.4.2 Cache division

In the proposed hybrid caching approach, a relative part $\lambda \in [0; 1]$ of the leased capacity for each provider is used as a reactive cache, for example applying the Least Recently Used (LRU) replacement strategy. The remaining part $(1 - \lambda)$ of the leased capacity is used for proactive placement. The value of λ is called the reactive ratio of the hybrid caching system. When a provider $p \in P$ leases a capacity of d_p bytes, then $\lambda \times d_p$ bytes are used for reactive caching, while the remaining $(1 - \lambda) \times d_p$ bytes are used for proactive placement.

Instead of using a fixed reactive ratio on every node, the reactive part of the leased capacity is uniformly distributed across the entire topology, proportional to the storage capacity of the nodes. In this way, the entire network is provided with reactive caching capacity to deal with unexpected popularity fluctuations, independently of the geographical distribution of the predicted request pattern. When $\lambda \times d_p$ bytes are used for reactive caching for provider $p \in P$, on every node $n \in N_{ISP}$, the number of bytes allocated for reactive caching for provider p can be calculated using equation (3.1).

$$c_n \times \frac{\lambda \times d_p}{\sum_{n' \in N_{ISP}} c_{n'}} \tag{3.1}$$

The remaining leased capacity of $(1-\lambda)\times d_p$ bytes are used for proactive placement. The allocation of this capacity, spread across the network, is based on the ILP model, described in Section 3.4.3. For each node $n \in N_{ISP}$, the total capacity available for proactive placement c_n^{pro} can be calculated using equation (3.2).

$$c_n^{pro} = c_n - \frac{c_n \times \lambda \times \sum_{p \in P} d_p}{\sum_{n' \in N_{ISP}} c_{n'}} \tag{3.2}$$

Using this hybrid approach, the geographical distribution of the allocation of proactive caching capacity is based on the predicted request pattern, allocating more capacity where more requests are expected. However, by uniformly distributing the reactive capacity, every area in the network is provided with some backup capacity to deal with possible errors in the request prediction or rapid popularity fluctuations.

3.4.3 Proactive placement

The ILP, used to model the considered problem, is a modified version of the ILP described in our previous work [1]. The modifications handle the difficulty in accurately predicting the exact time points at which a request will be sent. Therefore, while the ILP model in our previous work required the prediction of specific time points of requests, the modified ILP requires more realistic aggregated request predictions (i.e., the total number of predicted requests over the considered

period instead of exact timing information of each request). Furthermore, the current placement configuration is taken into account in order to limit the migration overhead.

In the remainder of this section, the input values, the decision variables, the objective functions and the constraints of the ILP are presented.

3.4.3.1 Input values

The objective of the proposed approach is to periodically compute a new caching configuration based on the estimation of content popularity and geographical distribution of requests for the next provisioning interval. As such, besides the characteristics of the network topology, the content catalog and the leased capacity for each provider, a prediction of the request pattern for the considered time interval is required by the algorithm at each reconfiguration step to determine a new content placement and server selection strategy. In addition to the general notations introduced in the previous section, we note $r_{o,d} \in \mathbb{N}$ as the predicted number of requests for content $o \in O$, originating from edge node $d \in N_E$ in the considered provisioning interval. $N^o \subseteq N_E$ represents the set of edge nodes requesting content $o \in O$ in the considered interval. To be able to take into account the current placement configuration, an additional notation $X_n \subseteq O$ is introduced for each node $n \in N$, representing the set of content items that are currently stored at node n.

3.4.3.2 Decision variables

A solution to the content placement problem is translated into binary decision variables $x_{n,o} \in \{0, 1\}$ defining if a node $n \in N$ is used to store content $o \in O$. In addition, auxiliary decision variables $z_{n,o,d} \in \{0, 1\}$ are introduced to represent the server selection strategy. These variables define if a node $n \in N$ is used to store content $o \in O$ to be delivered to edge node $d \in N_E$.

3.4.3.3 Objective function

Different optimization criteria have been considered in the literature [6, 7, 9]. Even though other metrics such as delivery delay minimization or link load minimization can easily be integrated in the problem formulation, in this chapter we focus on reducing the ISP network resource usage. As such, we define the optimal solution to the problem as the one minimizing the bandwidth usage inside the ISP network. While calculating the bandwidth usage, a weighting factor $\alpha \in [0; 1]$ can be used to define the importance of ingress link usage. In this way, the objective function can be tuned to purely optimize bandwidth usage or to focus on optimizing the hit ratio. Low values of α steer the ILP to minimize the bandwidth usage within the ISP network. In contrast, higher values of α result in minimizing the bandwidth

usage on the ingress link, yielding a maximization of the hit ratio. In this work, we focus on optimizing the total bandwidth usage, considering both the server link usage and the bandwidth usage inside the ISP network. Therefore, a value of $\alpha = 0.5$ is used, unless otherwise stated. The link weight ω_l of a link $l \in L$ is shown in equation (3.3).

$$\omega_l = \begin{cases} \alpha & \text{if } l \in L_S \\ 1 - \alpha & \text{if } l \in L_{ISP} \end{cases} \tag{3.3}$$

In order to minimize the total bandwidth usage in the ISP network, the bandwidth usage incurred by the video streaming sessions is modeled in the basic objective function (3.4). However, besides the video streaming sessions, the periodic content migrations significantly influence the total bandwidth usage. Therefore, this migration overhead can be taken into consideration in the objective function as well, based on the current storage configuration as shown in equation (3.5). Content that has to be placed at a specific node is fetched from the server node S. In the overhead-aware objective function, shown in equation (3.6), both the video streaming bandwidth and the migration overhead are taken into account. It can be seen that only the streaming bandwidth objective function (3.4) depends on predicted values (i.e., the predicted request intensities $r_{o,d}$). All remaining variables have known values. In the remainder of this chapter, we will refer to (3.4) and (3.6) as the basic objective function and the overhead-aware objective function, respectively, both of which are subject to minimization in the considered ILP.

$$BW_{str} = \sum_{n \in N} \sum_{o \in O} \sum_{d \in N^o} \sum_{l \in R_{n,d}} \omega_l \times r_{o,d} \times s_o \times z_{n,o,d} \tag{3.4}$$

$$BW_{mig} = \sum_{n \in N} \sum_{o \in O \setminus X_n} \sum_{l \in R_{S,n}} \omega_l \times s_o \times x_{n,o} \tag{3.5}$$

$$BW = BW_{str} + BW_{mig} \tag{3.6}$$

3.4.3.4 Constraints

Multiple constraints are considered to define the set of valid solutions to the considered optimization problem. First of all, auxiliary constraints are introduced to formalize the relationship between the x and z decision variables. The constraints presented in equation (3.7) and equation (3.8) specify that content $o \in O$ is stored at node $n \in N$ if and only if at least one edge node $d \in N_E$ requests o from n.

$$\forall n \in N, \forall o \in O, \forall d \in N_E : \quad z_{n,o,d} \leq x_{n,o} \tag{3.7}$$

$$\forall n \in N, \forall o \in O : \quad x_{n,o} \leq \sum_{d \in N_E} z_{n,o,d} \tag{3.8}$$

Table 3.2: Summary of the algorithm-specific notations.

Notation	Description
$\lambda \in [0;1]$	Reactive ratio.
$c_n^{pro} \in \mathbb{N}^+$	Proactive capacity of node $n \in N_{ISP}$.
$r_{o,d} \in \mathbb{N}$	Nr. of requests for $o \in O$ from $d \in N_E$.
$N^o \subseteq N_E$	Edge nodes requesting $o \in O$.
$X_n \subseteq O$	Content items stored at node $n \in N$.
$x_{n,o} \in \{0,1\}$	$o \in O$ placed on $n \in N$.
$z_{n,o,d} \in \{0,1\}$	$o \in O$ placed on $n \in N$ for $d \in N_E$.
$w_l \in [0;1]$	Weight of link $l \in L$.

A valid solution to the optimization problem is so that the caching space reserved for each content provider $p \in P$ is at most equal to the part of the leased capacity assigned for proactive content placement, while satisfying the storage capacity limitations. These constraints are modeled in equation (3.9) and equation (3.10), respectively.

$$\forall p \in P: \quad \sum_{n \in N_{ISP}} \sum_{o \in O_p} s_o \times x_{n,o} \leq (1-\lambda) d_p \tag{3.9}$$

$$\forall n \in N_{ISP}: \quad \sum_{o \in O} s_o \times x_{n,o} \leq c_n^{pro} \tag{3.10}$$

Finally, constraint equation (3.11) ensures that every request is served from exactly one location.

$$\forall o \in O, \forall d \in N^o : \sum_{n \in N} z_{n,o,d} = 1 \tag{3.11}$$

Periodically solving the ILP results in a storage profile represented by the values of $x_{n,o}$ and a server selection strategy represented by the values of $z_{n,o,d}$, which minimizes the objective function in equation (3.4) or equation (3.6), while satisfying the constraints in equations (3.9 – 3.11). Every request from edge node $d \in N_E$ for content $o \in O$ is served from node $n \in N$ where $z_{n,o,d} = 1$, using the shortest path $R_{n,d}$.

For the reader's reference, the algorithm-specific notations are summarized in Table (3.2).

3.5 Evaluation setup

To thoroughly evaluate the performance of the proposed approach, a topology based on the GÉANT network[3] is used, consisting of 23 nodes. As described

[3]GÉANT Project - http://www.geant.net

in Section 3.3.2, the considered VoD request trace contains 12 cities, which are mapped on 12 edge nodes ($N_E = \{E1, ..., E12\}$). One node is selected as the external server node S, storing all content of all content providers. The 10 remaining nodes are modeled as core nodes ($N_C = \{C1, ..., C10\}$). The resulting topology is shown in Figure 3.6. In this topology, shortest path routing based on hop count is applied.

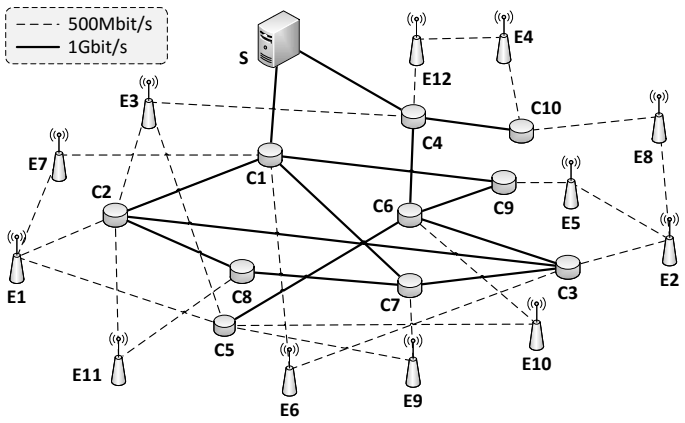


Figure 3.6: Evaluated GÉANT-based topology.

As preliminary evaluations have shown that the node capacities have limited influence on the performance of the approach, unless otherwise stated, the storage capacity of each core node was set high enough to be able to accommodate the leased capacity of all tenants throughout the evaluations. Concretely, for every core node $n \in V_C$, the capacity is defined as $c_n = \sum_{p \in P} d_p$. The capacity of the edge nodes is fixed to half of the capacity of the core nodes, i.e., $c_n = 0.5 \times \sum_{p \in P} d_p, \forall n \in V_E$. The bandwidth capacity of the links interconnecting core nodes and links connected to the server node is set to 1Gbit/s while all other links have a bandwidth capacity of 500Mbit/s.

Throughout the evaluations, the performance of the proposed approach is compared to a purely reactive approach. For the reactive approach, the LRU replacement strategy is applied, while the leased capacity is uniformly spread across the network, proportional to the node capacities. All requests are sent to the origin server S, applying reactive caching. Consequently, the reactive approach can result in a configuration where only parts of a movie are available at a given node. In contrast, the proactive approach either places an entire movie at a specific node or does not store it there at all.

Experiments have been performed for all of the 30 days in the VoD trace, while only evaluating the last 23 days (from February 13, 2010 to March 7, 2010). The first 7 days of the trace were used for obtaining the request prediction used in the

proactive approach and serve as a cache-warming phase for the reactive approach. To periodically determine the proactive placement configuration, the ILP is solved using the *IBM ILOG CPLEX Optimization Studio 12.4* solver.

3.6 Evaluation results

To characterize the performance of the proposed approach, multiple performance indicators are evaluated:

- **Hit ratio:** the relative amount of segment requests that could be served from within the ISP network (in %).
- **Average bandwidth usage:** the average bandwidth usage in the entire ISP network (in Mbit/s), including both the video streaming bandwidth and the content migration bandwidth induced by the proactive placement.
- **Migration overhead:** the total amount of data transfer induced by the proactive content placement (in Gbyte).
- **Average hop count:** the average number of links a segment crosses between its storage location and the requesting client. This metric indicates how close the relevant movies are stored to the end-users.

First, the different aspects of the proposed hybrid cache management approach are evaluated in Section 3.6.1. Next, the performance of our approach is compared to a purely proactive and a purely reactive cache management approach in Section 3.6.2.

3.6.1 Influence of the system parameters

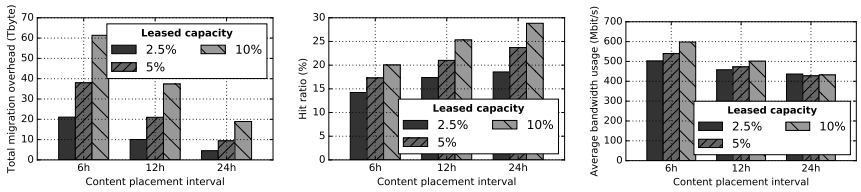
In this section, the optimal configuration of the proposed approach is determined, starting from a purely proactive approach ($\lambda = 0$) using the basic objective function (3.4) without blockbuster movie knowledge and gradually evaluating the added value of more complexity. Unless stated differently, in all of the evaluations, the content catalog has been uniformly distributed between two content providers, each leasing storage capacity to accommodate 2.5%, 5% or 10% of their total movie catalog.

3.6.1.1 Proactive placement frequency

The frequency at which a new proactive placement configuration is computed defines a trade-off between optimality and overhead. While more frequent reconfigurations allow the system to be more reactive with respect to changes in the request pattern, this comes at the cost of more frequent content migrations. Figure 3.7

shows the influence of the length of the time interval between subsequent content placements on multiple performance indicators. For these evaluations, a purely proactive approach has been applied using the basic objective function without blockbuster movie knowledge. As could be expected, Figure 3.7a shows that the content migration overhead strongly increases when more frequent reconfigurations are performed, independently of the leased capacity.

Remarkably however, less frequent reconfigurations also result in a higher performance in terms of hit ratio and average bandwidth usage, as can be seen in Figure 3.7b and Figure 3.7c, respectively. This counter-intuitive result can be explained by the characteristics of the VoD request trace and the applied prediction strategy, discussed in Section 3.3.3. As was shown in Figure 3.4a, the prediction accuracy significantly decreases when predictions are made for shorter periods of time. In the case of more frequent reconfigurations, the low prediction accuracy results in more requests that have to be fetched from the origin server, resulting in longer routing paths, which lead to higher bandwidth usage and lower hit ratio. Similar observations can be made for the average hop count, which is on average 4.87% lower when placements are performed every 24h compared to every 6h (graphs omitted due to space limitations). Given the performance degradation in terms of prediction accuracy for periods longer than 24h (as described in Section 3.3.3) and the strong diurnal pattern in the request trace, reconfiguration frequencies lower than once every 24h have not been considered in the analysis.



(a) Influence of the proactive placement frequency on the total migration overhead.

(b) Influence of the proactive placement frequency on the hit ratio.

(c) Influence of the proactive placement frequency on the average bandwidth usage in the network.

Figure 3.7: Influence of the proactive placement frequency in a purely proactive scenario using the basic objective function without blockbuster movie knowledge.

The influence of the proactive placement frequency on the average time needed by the CPLEX solver to solve the ILP is presented in Figure 3.8. The error bars show the standard deviation. It can be seen that the execution time significantly increases with the content placement interval, as an increasing number of requests results both in a higher number of decision variables and constraints and an increased complexity for the objective function of the ILP. The high standard deviation values are due to the high fluctuation of the request intensity over time

as shown in Figure 3.3. However, it can be seen that in the considered scenario, the average time needed to solve the ILP is reasonable compared to the placement interval (e.g., 15s every 24h on average for a leased capacity of 10%).

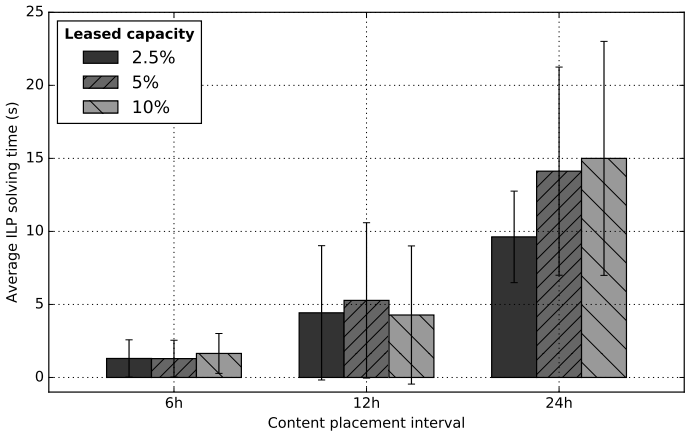


Figure 3.8: Influence of the proactive placement frequency on the average time needed to solve the ILP.

In the remainder of this chapter, we assume that the proactive placement is performed every 24h given that this frequency results in the best performance. Furthermore, this allows the ISP to perform content migrations in off-peak hours (e.g., at night).

3.6.1.2 Overhead-aware placement

The benefits of adding overhead-awareness to the proactive placement are evaluated in a purely proactive scenario without blockbuster movie knowledge. Figure 3.9 shows the relative performance of the approach using the overhead-aware objective function compared to the performance of the approach using the basic objective function. It can be seen that taking into account the migration overhead in the placement decisions drastically reduces the migration overhead by 37.49% on average. Furthermore, the reduction in terms of migration overhead results in 3.33% less bandwidth usage on average. Given that, on average, the bandwidth introduced by the content migration only amounts to about 10% of the total bandwidth usage, this bandwidth reduction can be fully attributed to the reduced migration bandwidth. The deviation of the bandwidth usage introduced by the streaming sessions is limited to less than 1%. With a relative gain of 1.37% and 0.3% respectively, the influence on the performance in terms of hit ratio and average hop count is negligible. Furthermore, due to its higher complexity, using the overhead-aware

Table 3.3: Stability of the content popularity.

Amount*	**Stability**
2.5%	61.28%
5%	57.41%
10%	53.89%
20%	51.41%
100%	37.30%

*Relative to the total number of requested contents on given day.

objective function results in a (limited) increase of 6.99% in terms of the average time needed to solve the ILP.

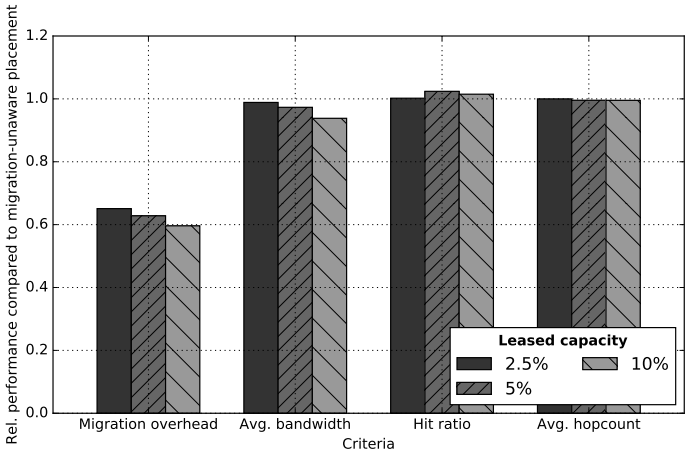


Figure 3.9: Influence of overhead-awareness in proactive content placement on different evaluation criteria.

As can be observed in Figure 3.9, the benefits of introducing overhead awareness to the system increase when more capacity is leased. The explanation for this observation can be found in the shifting characteristics of the VoD request trace. Based on Figure 3.5, we can calculate the average percentage of the x most popular content on a given day that still belongs to the x most popular content on the next day with x being equal to 2.5%, 5%, 10%, 20% or 100% of the total number of content requested on a given day. The resulting values, referred to as the *stability* of the content popularity, are shown in Table 3.3. It can be seen that the stability of the popularity decreases when a higher amount of content is considered.

When more capacity is leased, content items with a lower popularity can also

be cached proactively. However, as demonstrated, these items change more frequently than the popular ones. By adding overhead-awareness to the system, the placement algorithm can balance the bandwidth gain of placing the less popular content in the network against the cost of migrating the content. For popular content, the streaming bandwidth reduction exceeds the migration cost, while this may no longer be the case for less popular content. Therefore, overhead-awareness adds more value when less popular content can be placed in the network.

In the remainder of the evaluations, the overhead-aware objective function is used to calculate the proactive placement configuration.

3.6.1.3 Blockbuster movie knowledge

As was shown in Section 3.3.3, the accuracy of the popularity prediction can be increased based on the availability of exogenous information about blockbuster movies. The influence of this information on the performance of the proposed approach in terms of hit ratio is shown in Figure 3.10. As expected, the perceived hit ratio increases with the number of known blockbuster movies. It is also interesting to see that the performance increase in terms of hit ratio exceeds the increased prediction accuracy. For example, when the leased capacity is equal to 5% of the total content catalog, knowledge about 5 blockbuster movies leads to a relative increase of 3.97% in terms of prediction accuracy (Figure 3.4b), while the relative improvement in terms of perceived hit ratio amounts to 17.22% (i.e., an absolute increase of 4.18% compared to the original hit ratio of 24.27%, see Figure 3.10).

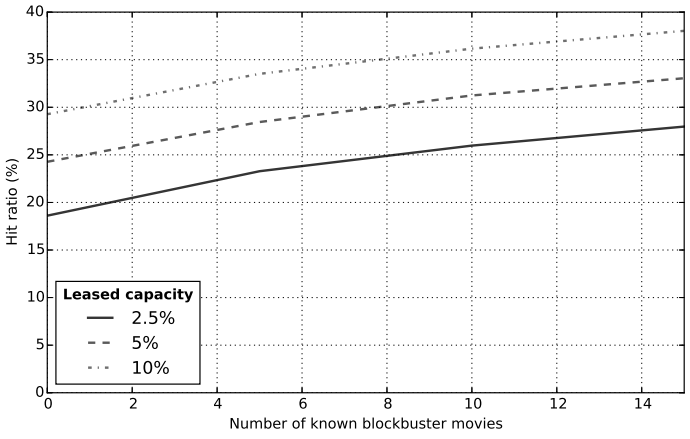


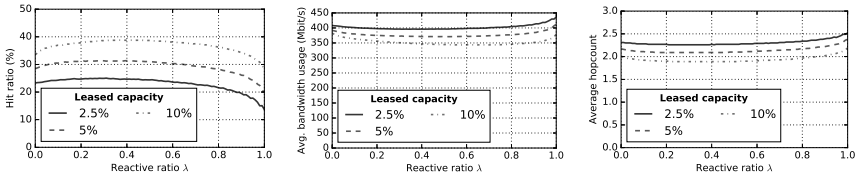
Figure 3.10: Influence of blockbuster movie knowledge on the performance in terms of hit ratio.

To clarify this observation, it is important to note that the goal of the proactive placement is to minimize the bandwidth usage. Therefore, storing duplicates

of popular content can be preferred over storing as much unique content items as possible, causing the perceived hit ratio to be lower than what could be achieved based on the prediction accuracy. This means that some of the less popular content items are not selected by the placement algorithm, even though they could be cached given the value of the prediction accuracy metric. The information about blockbuster movies can be used to derive the real popularity of some of these content items, resulting in a higher hit ratio and this without affecting the prediction accuracy.

In terms of average bandwidth usage and average hop count, a relative performance increase of 5.79% and 5.65% is achieved respectively, while the migration overhead is rather unaffected (decrease of 0.12%). In what follows, unless otherwise stated, it is assumed that the system is informed about the 5 most popular blockbuster movies by an external source, as described in Section 3.3.3.

3.6.1.4 Hybrid cache division



(a) Influence of the reactive ratio λ on the hit ratio.

(b) Influence of the reactive ratio λ on the average bandwidth usage.

(c) Influence of the reactive ratio λ on the average hop count.

Figure 3.11: Influence of the reactive ratio λ on the performance of the hybrid caching approach.

Up to now, a purely proactive approach has been followed. In Section 3.4, it was argued that applying a hybrid caching approach can significantly improve the performance of the system. To evaluate the influence of the cache division, the reactive ratio λ has been varied between $\lambda = 0.0$ (i.e., a purely proactive approach) and $\lambda = 1.0$ (i.e., a purely reactive approach) in a scenario where knowledge about 5 blockbuster movies is available for the request prediction and the overhead-aware objective function is used. The proactive placement configuration is recalculated every 24h.

Figure 3.11 shows the influence of the reactive ratio λ on the hit ratio, the average bandwidth usage and the average hop count of the proposed approach. It can be seen that, in the majority of the cases, both the purely proactive and the purely reactive approach are outperformed by the hybrid scheme in terms of hit ratio, average bandwidth usage and average hop count (i.e., for 79%, 94% and

78% of the λ values, respectively). Given that the lease constraint (3.9) of the placement ILP limits the amount of placed content to the size of $(1 - \lambda)d_p$ for each provider $p \in P$, the migration overhead decreases linearly with increasing reactive ratio λ (graph omitted due to space limitations).

The reactive ratio that gives the optimal performance in the considered scenario depends on the considered metric. However, as can be seen in Figure 3.11, a wide range of ratios has a performance close to the performance of the optimal ratio for each of the metrics. To be able to select a single ratio, the different metrics should be optimized simultaneously. Therefore, for each evaluated metric M and each reactive ratio λ, we define the deviation from the optimum δ_λ^M as shown in equation (3.12). It is calculated as the ratio between the performance of reactive ratio λ in terms of metric M, denoted as ϕ_λ^M, and the optimal performance for that metric, ω^M. ω^M is defined as the maximum or minimum value $\phi_{\lambda'}^M, \forall \lambda' \in [0; 1]$, depending on whether metric M should be maximized or minimized respectively.

$$\delta_\lambda^M = \left| 1 - \frac{\phi_\lambda^M}{\omega^M} \right| \tag{3.12}$$

For each reactive ratio λ the average value of δ_λ^M for the considered metrics (M_1: hit ratio, M_2: average bandwidth usage and M_3: average hop count) can be calculated as $\Delta_\lambda = \frac{1}{3}\left(\delta_\lambda^{M_1} + \delta_\lambda^{M_2} + \delta_\lambda^{M_3}\right)$, representing how close the performance is to the optimal performance on average. Figure 3.12 shows the average deviation from the optimum for different amounts of leased capacity. It can be seen that the optimal ratio slightly increases with an increasing amount of leased capacity. This can be explained by the growing performance of LRU for bigger caches and the reduced stability of the content popularity for less popular content. On average, the optimal performance is achieved with a reactive ratio $\lambda = 0.41$.

3.6.1.5 Reactive ratio adaptation

As described in Section 3.1, the performance of the proactive placement approach strongly depends on the quality of the popularity prediction. Given that the prediction accuracy fluctuates over time, the optimal reactive ratio λ is also subject to change. Therefore, changing the reactive ratio over time might be considered. For example, Figure 3.13 shows the optimal reactive ratio λ over time at a granularity of 6h for different amounts of leased capacity in order to maximize the hit ratio. It can be seen that the optimal ratio indeed strongly fluctuates over time without following any well-defined pattern.

Figure 3.14 shows the relative hit ratio when using a fixed reactive ratio $\lambda = 0.41$ compared to using the optimal ratios shown in Figure 3.13 at a granularity of 6h. It can be seen that the hit ratio using a fixed reactive ratio is close to the optimal adaptive performance, with an average deviation of only 2.90%. Furthermore, it is

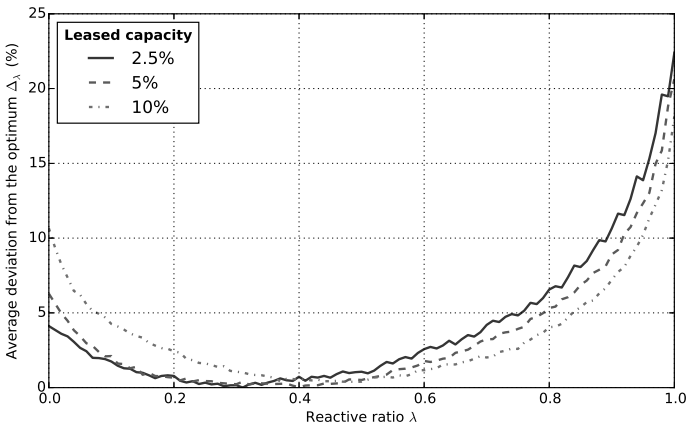


Figure 3.12: Influence of the reactive ratio λ on the average deviation from the optimum.

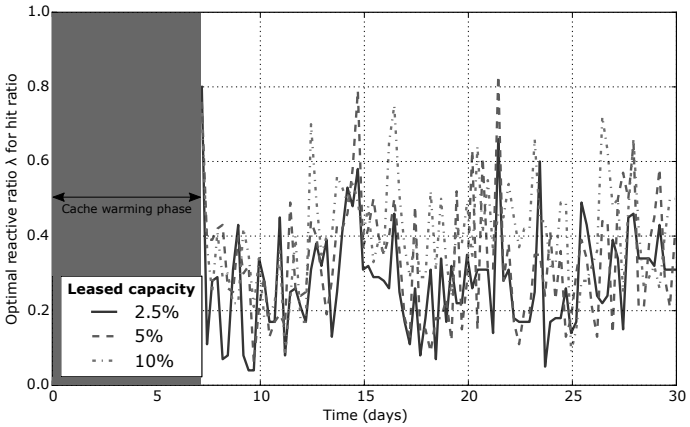


Figure 3.13: Optimal reactive ratio λ over time.

important to note that the optimal performance is a theoretical optimum that would require to be able to determine the optimal reactive ratio at any point in time.

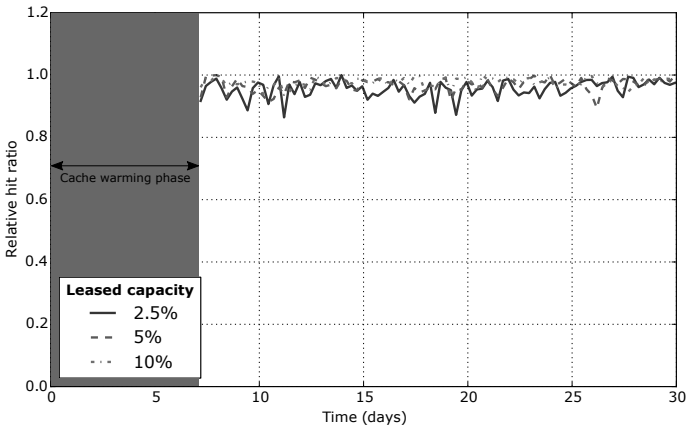


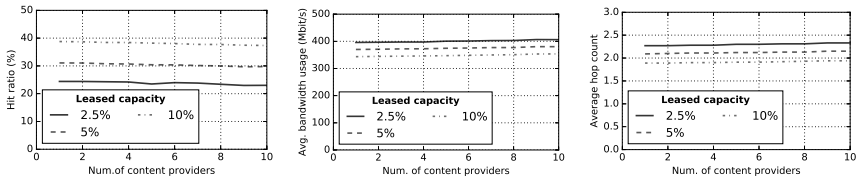
Figure 3.14: Relative hit ratio with $\lambda = 0.41$ compared to an adaptive reactive ratio.

Adaptively changing the reactive ratio λ would introduce a lot of added complexity to the system, as it requires to be able to monitor the accuracy of the request prediction. Furthermore, given the noisy pattern shown in Figure 3.13, the accuracy of predicting the optimal ratio would be limited, resulting in a lower performance gain than theoretically possible. In addition, changing the reactive ratio demands a reconfiguration of both the reactive and the proactive cache at every single node in the network, again introducing additional migration overhead. Given this high level of complexity and the limited theoretical performance increase that could be achieved in the considered scenario, a fixed reactive ratio is proposed in practice.

3.6.1.6 Number of tenants

Up to now, all the evaluations have been performed for two content providers. To show the applicability of the proposed approach in the general case, evaluations have been performed for multiple tenants leasing storage capacity from the ISP. For this purpose, the VoD movie catalog of 5644 movies has been uniformly split amongst the tenants (i.e., each movie in the catalog has been randomly assigned to one of the tenants with equal probability). Even though the characteristics of the request trace and the content catalog are unchanged and the total amount of leased capacity across the tenants remains the same in all the evaluations (i.e., independent of the number of tenants), the results of the proactive placement approach are influenced by the multi-tenancy. Given the leased capacity constraint (3.9) for each provider, a fixed amount of content items has to be stored for each provider

to meet their capacity requirements, even though other providers might have more popular content. In contrast, in a scenario with a single content provider, only the globally most popular content will be proactively placed. Figure 3.15 shows the influence of the number of tenants on the hit ratio, the average bandwidth usage and the average hop count. It can be seen that the influence of increasing the number of tenants from 1 to 10 is limited to a relative performance degradation of 4.54%, 2.78% and 2.72%, respectively. These values are in line with the degree of performance degradation obtained in both the purely proactive and reactive cases. The results show that the proposed approach can be applied with a fixed parameter configuration which is independent of the number of tenants, without incurring any significant performance degradation.



(a) Influence of the number of content providers on the hit ratio.

(b) Influence of the number of content providers on the average bandwidth usage.

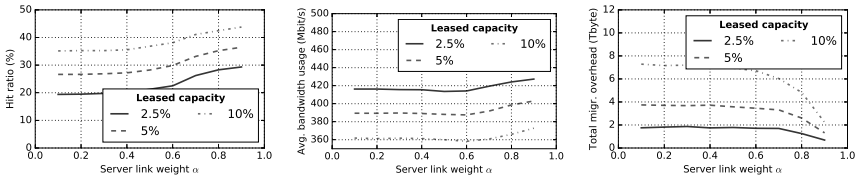
(c) Influence of the number of content providers on the average hop count.

Figure 3.15: Influence of the number of content providers on the performance of the hybrid caching approach.

3.6.1.7 Server link weight

In all of the above evaluations, the server link weight α has been fixed to 0.5 in order to optimize the bandwidth usage. As explained in Section 3.4.3, the value of α can be used to steer the placement decisions between optimizing the bandwidth usage and optimizing the hit ratio. For the sake of completeness, the influence of the server link weight α on the performance of the proposed approach is evaluated.

Figure 3.16 shows the influence of the server link weight α on the performance of the hybrid caching approach with a reactive ratio of $\lambda = 0.41$, using the overhead-aware objective function and a request prediction without knowledge of blockbuster movies. The proactive placement is performed every 24h. Using a value of $\alpha > 0.5$ steers the ILP to minimize the bandwidth usage on the server ingress link. Minimizing the server link usage directly results in a higher hit ratio, as can be seen in Figure 3.16a. In terms of bandwidth usage, Figure 3.16b shows a clear increase for the values of $\alpha > 0.5$. When the focus is given to the ingress link bandwidth, the algorithm prefers storing more unique movies in the network instead of duplicating the most popular ones. This results in longer routing paths for



(a) Influence of the server link weight α on the hit ratio.

(b) Influence of the server link weight α on the average bandwidth usage.

(c) Influence of the server link weight α on the migration overhead.

Figure 3.16: Influence of the server link weight α on the performance of the hybrid caching approach.

the popular content, introducing additional bandwidth usage. Finally, Figure 3.16c shows that for large values of α, the migration overhead is significantly reduced. This behavior is due to the fact that the overhead-aware objective function is used. As all migrated content originates from the server node, all migration overhead is routed over a server ingress link, having a strong impact on the objective of the ILP. Placing content inside the ISP network can decrease the streaming bandwidth usage, but for large values of α, the bandwidth inside the ISP network is not decisive. Consequently, large values of α will steer the algorithm to migrate less content items.

3.6.1.8 Capacity limitations

Up to now, all evaluations have been performed using over-provisioned node capacities, i.e., sufficiently high to allocate all of the leased capacity on a single core node. This gave the algorithm complete freedom on how to distribute the capacity. To demonstrate the general applicability of the proposed approach, simulations have been performed in a scenario where the total storage capacity available in the network is equal to the capacity leased by all of the tenants. As with the previous evaluations, the capacity of the edge nodes is set to half of the capacity of the core nodes. In this configuration, the algorithm has less freedom to decide how to place the content given that the space available at each node is limited.

Figure 3.17 shows the performance of the proposed approach when two tenants lease storage capacity in a scenario with limited node capacities, relative to the scenario with over-provisioned nodes. It can be seen that, due to the restricted freedom of the algorithm, the average bandwidth usage is slightly increased by 2.67%. Furthermore, the limited capacity causes the content to be stored further away from the end-user, resulting in a slight increase of 3.85% in terms of average hop count. However, it is interesting to note that the number of duplicated content items is reduced, resulting in the average hit ratio being increased by 2.85%.

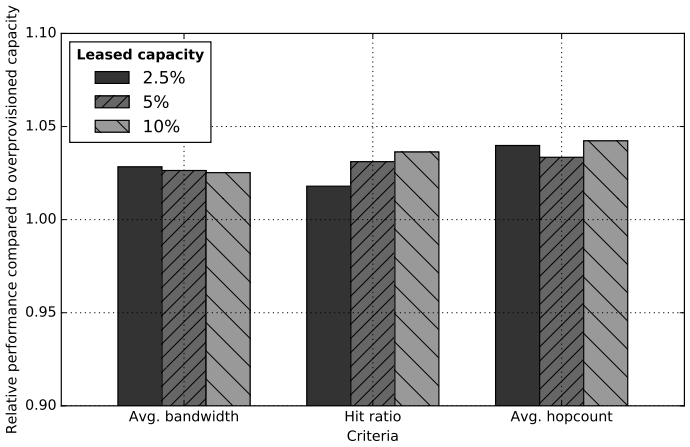


Figure 3.17: Relative performance of the proposed approach in a scenario with limited capacity compared to the scenario with over-provisioned capacity.

3.6.2 Performance comparison

In this section, we compare the performance of the proposed approach to a purely proactive and reactive scheme based on the preferred parameter configuration (i.e., the one that optimizes the performance of the hybrid scheme) derived from the analysis in Section 3.6.1:

- Reconfiguration interval of 24h
- Overhead-aware objective function
- Fixed reactive ratio $\lambda = 0.41$

In this section, we consider a scenario where no knowledge about blockbuster movies is available and a scenario where 5 blockbuster movies are known *a priori*. In these evaluations, two content providers lease some caching capacity from the ISP with over-provisioned node capacities.

Figure 3.18 shows the performance of the proposed approach, relative to a purely reactive approach in a scenario where no information about blockbuster movies is available. It can be seen that the benefits of the proposed approach are higher for lower amounts of leased capacity. This can be explained by the combination of the increasing performance of the LRU cache replacement strategy for larger cache sizes and the decreasing stability of the popularity for higher amounts of leased capacity, as described in Section 3.6.1.2. For the evaluated VoD scenario and topology, the relative performance increase on average amounts to 5.35%, 42.96% and 8.15% in terms of average bandwidth usage, hit ratio and average hop count respectively.

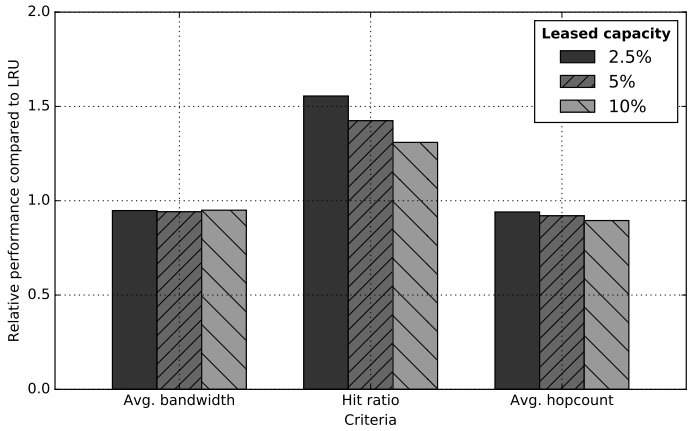


Figure 3.18: Relative performance of the proposed approach compared to a purely reactive approach when no blockbuster knowledge is available.

In the same scenario, the performance of the proposed approach is compared to a purely proactive approach in Figure 3.19. As opposed to the comparison with a reactive approach, the performance gain compared to a proactive approach increases with the amount of leased capacity. Again, this can be explained by the increasing performance of the LRU replacement strategy and the decreasing stability of the content popularity: the reactive part of the hybrid cache covers the decreasing performance of the proactive placement strategy in these scenarios. The average performance increases amount to 7.36%, 18.78% and 5.63% in terms of average bandwidth usage, hit ratio and average hop count, respectively, with 39.19% less migration overhead.

When information about 5 blockbuster movies is available on a daily basis, the same trends can be seen (graphs omitted due to space limitations). Compared to a purely reactive approach, the average performance increase in terms of average bandwidth usage, hit ratio and average hop count is equal to 9.26%, 44.34% and 8.46%, respectively. Compared to a purely proactive approach, relative performance increases of 5.89%, 10.58% and 3.94% are achieved in terms of average bandwidth usage, hit ratio and average hop count, respectively, with 39.63% less migration overhead.

3.7 Conclusions

In this chapter, we presented a hybrid cache management approach for ISP networks in a scenario where multiple content providers lease caching capacity. In the proposed approach, a part of the leased capacity is uniformly allocated across

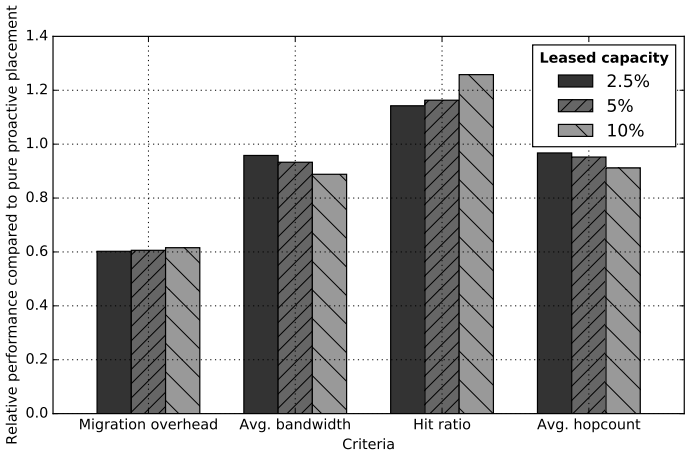


Figure 3.19: Relative performance of the proposed approach compared to a purely proactive approach when no blockbuster knowledge is available.

the network and equipped with a reactive cache replacement strategy to handle inherent inaccuracies when predicting the content popularity. The allocation of the remaining caching capacity is periodically reconfigured based on predictions of the content popularity and the geographical distribution of requests.

To characterize the performance of the proposed approach in a realistic scenario, a request trace of the VoD service of a leading European telecom operator has been applied. While general characteristics of this trace, such as its popularity curve and diurnal pattern, are comparable to generally accepted popularity models, it contains real-life information about the geographical distribution of content popularity and underlying relationships between multiple content items, which is often nonexistent in synthetic models.

Thorough evaluations have shown that a hybrid cache management approach can combine the benefits of both a reactive approach and the purely proactive management approach proposed in previous work [1, 4], outperforming both approaches in terms of multiple performance indicators for a wide range of reactive ratios. After optimizing the reactive ratio over multiple scenarios for the evaluated VoD use-case and topology, the hybrid cache management approach can increase the hit ratio by 18.78% and 42.96% on average, compared to the purely proactive and the purely reactive approach using the LRU cache replacement strategy, respectively. In terms of average bandwidth usage, the reduction amounts to 7.36% and 5.35%, respectively. Furthermore, content is shown to be stored closer to the end-users, indicated by the average hop count being reduced with 5.63% and 8.15%, respectively. Compared to the purely proactive approach, the hybrid caching approach introduces 39.19% less migration overhead.

Finally, the potential benefits of adaptively changing the cache division ratio over time in order to react to changes in the prediction accuracy have been investigated. It was shown that the theoretical performance gain with respect to a static reactive ratio in terms of hit ratio is negligible compared to the required level of added complexity.

Acknowledgment

Maxim Claeys is funded by grant of the Agency of Innovation by Science and Technology in Flanders (IWT). This work was partly funded by Flamingo, a Network of Excellence project (318488) supported by the European Commission under its Seventh Framework Programme.

References

[1] M. Claeys, D. Tuncer, J. Famaey, M. Charalambides, S. Latré, G. Pavlou, and F. De Turck. *Proactive multi-tenant cache management for virtualized ISP networks*. In Proceedings of the International Conference on Network and Service Management (CNSM), pages 82–90, 2014.

[2] Cisco. *Cisco Visual Networking Index: Forecast and methodology, 2014-2019*. Technical report, Cisco, 2015.

[3] W. Jiang, R. Zhang-Shen, J. Rexford, and M. Chiang. *Cooperative content distribution and traffic engineering in an ISP network*. In Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS), pages 239–250, 2009.

[4] M. Claeys, D. Tuncer, J. Famaey, M. Charalambides, S. Latré, F. De Turck, and G. Pavlou. *Towards multi-tenant cache management for ISP networks*. In Proceedings of the European Conference on Networks and Communications (EuCNC), pages 1–5, 2014.

[5] S. Asur and B. A. Huberman. *Predicting the future with social media*. In Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, volume abs/1003.5, pages 492–499, 2010.

[6] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, and K. K. Ramakrishnan. *Optimal content placement for a large-scale VoD system*. In Proceedings of the International Conference on Emerging Networking Experiments and Technologies (CoNEXT), pages 1–12, nov 2010.

[7] A. Sharma. *Distributing content simplifies ISP traffic engineering*. In Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS), pages 229–242, 2013.

[8] V. Sourlas, P. Flegkas, L. Gkatzikis, and L. Tassiulas. *Autonomic cache management in information-centric networks*. In Proceedings of the IEEE Network Operations and Management Symposium (NOMS), pages 121–129, 2012.

[9] N. Laoutaris, O. Telelis, V. Zissimopoulos, and I. Stavrakakis. *Distributed selfish replication*. IEEE Transactions on Parallel and Distributed Systems (TPDS), 17(12):1401–1413, 2006.

[10] S. Borst, V. Gupta, and A. Walid. *Distributed caching algorithms for content distribution networks*. In Proceedings of the IEEE International Conference on Computer Communications (INFOCOM), pages 1–9, 2010.

[11] J. Dai, Z. Hu, B. Li, J. Liu, and B. Li. *Collaborative hierarchical caching with dynamic request routing for massive content distribution*. In Proceedings of the IEEE International Conference on Computer Communications (INFOCOM), pages 2444–2452, 2012.

[12] T. Wauters, J. Coppens, F. De Turck, B. Dhoedt, and P. Demeester. *Replica placement in ring based content delivery networks*. Computer Communications, 29(16):3313–3326, 2006.

[13] M. R. Korupolu and M. Dahlin. *Coordinated placement and replacement for large-scale distributed caches*. IEEE Transactions on Knowledge and Data Engineering (TKDE), 14(6):1317–1329, 2002.

[14] V. Valancius, B. Ravi, N. Feamster, and A. C. Snoeren. *Quantifying the benefits of joint content and network routing*. In Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS), pages 243–254, 2013.

[15] B. Frank, I. Poese, G. Smaragdakis, S. Uhlig, and A. Feldmann. *Content-aware traffic engineering*. ACM SIGMETRICS Performance Evaluation Review, feb 2012.

[16] I. Baev and R. Rajaraman. *Approximation algorithms for data placement problems*. SIAM Journal on Computing, pages 1–18, 2008.

[17] T. Bekta, J.-F. Cordeau, E. Erkut, and G. Laporte. *Exact algorithms for the joint object placement and request routing problem in content distribution networks*. Computers & Operations Research, 35(12):3860–3884, dec 2008.

[18] N. Laoutaris, V. Zissimopoulos, and I. Stavrakakis. *Joint object placement and node dimensioning for Internet content distribution*. Information Processing Letters, 89(6):273–279, 2004.

[19] N. Laoutaris, V. Zissimopoulos, and I. Stavrakakis. *On the optimization of storage capacity allocation for content distribution*. Computer Networks, 47(3):409–428, 2005.

[20] N. Kamiyama, T. Mori, R. Kawahara, S. Harada, and H. Hasegawa. *ISP-operated CDN*. In Proceedings of the IEEE International Conference on Computer Communications (INFOCOM) Workshops, pages 49–54, 2009.

[21] K. Cho, H. Jung, M. Lee, D. Ko, T. Kwon, and Y. Choi. *How can an ISP merge with a CDN?* IEEE Communications Magazine, 49(October):156–162, 2011.

[22] M. Wichtlhuber, R. Reinecke, and D. Hausheer. *An SDN-based CDN/ISP collaboration architecture for managing high-volume flows*. IEEE Transactions on Network and Service Management (TNSM), 12(1):48–60, mar 2015.

[23] D. Tuncer, M. Charalambides, R. Landa, and G. Pavlou. *More control over network resources: an ISP caching perspective*. In Proceedings of the International Conference on Network and Service Management (CNSM), pages 26–33, 2013.

[24] M. K. Mukerjee, D. Naylor, J. Jiang, D. Han, S. Seshan, and H. Zhang. *Practical, real-time centralized control for CDN-based live video delivery*. ACM SIGCOMM Computer Communication Review, 45(5):311–324, 2015.

[25] N. Garg and J. Konemann. *Faster and simpler algorithms for multicommodity flow and other fractional packing problems*. SIAM Journal on Computing, 37(2):630–652, 2007.

[26] Y. Koren. *Factorization meets the neighborhood: a multifaceted collaborative filtering model*. In Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), pages 426–434, 2008.

[27] Y. Zhou, L. Chen, C. Yang, and D. M. Chiu. *Video popularity dynamics and its implication for replication*. IEEE Transactions on Multimedia, 17(8):1273–1285, 2015.

[28] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng. *Understanding user behavior in large-scale video-on-demand systems*. ACM SIGOPS Operating Systems Review, 40(4):333–344, 2006.

4

Cooperative Announcement-based Caching for Video-on-Demand Streaming

M. Claeys, N. Bouten, D. De Vleeschauwer, W. Van Leekwijck, S. Latré and F. De Turck.

Published in Transactions on Network and Service Management, June 2016.

⋆ ⋆ ⋆

In the previous chapter it was shown that a hybrid cache management approach can outperform purely proactive content placement by providing reactivity to unexpected changes in the request pattern. However, to further reduce the network load, the design of appropriate cache replacement algorithms is of utmost importance. Based on the fact that, typically, a video stream is temporally segmented into smaller chunks that can be accessed and decoded independently, cache replacement strategies have been developed that take advantage of this temporal structure in the video. In this chapter, two caching strategies are proposed that additionally take advantage of the phenomenon of binge watching, where users watch multiple consecutive episodes of the same series, which is reported by recent user behavior studies to become the everyday behavior. Taking into account this information allows us to predict future segment requests, even before the video playout has started. Two strategies are proposed, both with a different level of coordination between the caches in the network. Using a Video-on-Demand (VoD) request trace

based on binge watching user characteristics, the presented algorithms have been thoroughly evaluated in multiple network topologies with different characteristics, showing their general applicability. It was shown that in a realistic scenario, the proposed election-based caching strategy can outperform the state-of-the-art by 20% in terms of cache hit ratio while using 4% less network bandwidth.

4.1 Introduction

Traditionally, channel-based networks (e.g., satellite, cable networks) were used to distribute linear TV, benefiting from the broadcast nature of these networks. However, while in linear TV the amount of video traffic is proportional to the number of channels, in an on-demand system the traffic is proportional to the number of users. This leads to a major increase in capacity requirements for on-demand video delivery on the backbone. By deploying intermediary caches, closer to the end-users, popular content can be served faster and without increasing backbone traffic. Since video interests are shared between users and their viewing interests overlap in time, deploying these caches can substantially reduce the required capacity by taking advantage of these overlapping sessions. Caching networks (e.g., Content Delivery Networks (CDNs)) were traditionally used to deliver web content in a scalable way and reduce latency by temporally storing part of the content in a network of caches close to the end-users. However, the use of caching in a streaming environment differs from the traditional web-based caching, since the objects are typically much larger and are to be consumed directly after they have been requested. Therefore, video delivered over caching networks is typically temporally segmented into smaller chunks that can be accessed and decoded independently.

Designing an appropriate replacement strategy for such caching networks is of utmost importance to achieve high caching efficiency and reduce the network load. Most caching algorithms are based on observations and take into account the recency or frequency of requests to calculate the rank of each of the cached objects. When caching temporally segmented video, as is the case in HTTP-based streaming systems, the caching algorithm can also take into account the temporal structure of the video. That is, when the streaming application requests the n-th segment at a specific point in time, the caching strategy can take advantage of the knowledge that with high likelihood the (n+1)-th and subsequent segments will be consumed shortly after this segment. Therefore, when multiple streaming sessions request the same segmented content, the caching strategy can take advantage of this knowledge to keep the segments in cache that are to be consumed in the near future.

Caching strategies can be further improved by taking into account additional information, such as content popularity and regional differences. The optimal caching algorithm for an isolated cache (known as Belady's MIN [1]) takes ad-

vantage of the knowledge on future requests to discard the objects for which the next request occurs the furthest in the future. In a real system, this information on future requests is not directly available. However, studies[1] on user behavior for Over-The-Top (OTT) streaming services report that respectively 88% and 70% of the Netflix and Hulu Plus users stream three or more consecutive episodes of the same TV show (referred to as *binge watching*). Furthermore, binge watching is reported to become the everyday behavior with users streaming on average 2.32 episodes per sitting, resulting in about 57% of the streaming sessions that could be announced in advance [2]. Taking into account these announcements allows us to predict future segment requests and thus approximating the theoretical optimal caching strategy. Furthermore, service providers could give incentives (e.g., discounts, higher resolution video (4K)) to users when they announce their streaming sessions in advance.

In this chapter, a novel caching strategy is proposed that takes advantage of the inferred knowledge of future segment requests when streaming segmented video and additionally exploits the added knowledge when streaming clients announce the videos that will be streamed in the near future. Using this additional information, we are able to estimate the future reuse times of the segmented content. This allows us to increase the caching efficiency compared to using only reuse time predictions based on the structure of the segmented video. Two caching strategies are proposed, each applying a different type of cooperation inside the caching network. The first approach, based on the strategy presented in our previous work [3], applies a simple threshold-based cascading strategy to provide a basic level of coordination. The second proposal elaborates on this approach by adding a more advanced election-based coordination strategy. Furthermore, compared to our previous work, experiments have been performed using different network topologies and request traces, based on recent analysis of user behavior in a Video-on-Demand (VoD) scenario.

The main contributions of this chapter are threefold. First, we propose a novel election-based caching strategy that outperforms the current state-of-the-art strategies by including knowledge on segmented video streaming and announced future video requests. Second, we use user behavior models to emulate video interruptions and evaluate their impact on the proposed and state-of-the-art caching strategies. Finally, the proposed strategy is thoroughly evaluated in a distributed caching scenario on two types of network topology.

[1] Nielsen - http://www.nielsen.com/us/en/insights/news/2013/binging-is-the-new-viewing-for-over-the-top-streamers.html

4.2 Related work

4.2.1 Cache replacement strategies

Web caching was introduced as a way to save traffic and minimize the perceived delay on the Internet by storing data closer to the end-user. Many cache replacement algorithms have been proposed in the past and stem from replacement strategies for both web proxy caching and CPU caching. Least Recently Used (LRU) is based on the recency of the cached objects and discards the least recently used items first. This type of caching algorithms is known to attach too much importance to unpopular objects, awarding them the highest rank upon a single request. Least Frequently Used (LFU) determines the number of requests for an object over a period of time and discards the least frequently used objects first. This type of caching algorithms is known to attach as much importance to ancient as to recent history. To account for this, dynamic aging factors are introduced to improve the performance in LFU Dynamic Aging (LFU-DA). Adaptive Replacement Cache (ARC) balances between LRU and LFU by using more complex algorithms to determine the rank [4].

The aforementioned replacement strategies are widely used in web caches. Caching for video streaming differs from caching web content in a number of ways. First, video objects are typically much larger than traditional web objects. Second, there is a difference in popularity evolution for streaming videos compared to web content. Furthermore, in contrast to web objects, having only the beginning of a video allows a video application to already start playout while the download continues [5]. This has led to the adoption of segment-based caching of multimedia streams. These segments can have variable size, e.g., smaller segments at the start of the video to decrease start-up delays. *Wu et al.* define three segmentation approaches for proxy caching: fixed length segments, pyramid with exponentially increasing segment sizes and skyscraper where sizes increase slower compared to the pyramid scheme by repeating each layer n times [6, 7]. The authors conclude that segmentation approaches are particularly effective when the cache size is limited, media popularity changes over time, requests are spread over a large number of media objects and when the media file size and library is large.

Chen et al. propose a streaming proxy system called *Hyper Proxy*, which uses active prefetching for in-time delivery of uncached segments to address the proxy jitter problem [8]. This work was extended by adding segmentation strategies based on object popularity and by using predictions of future segment requests to preload uncached segments [9]. In previous work, a proactive cache management system for multi-tenant CDNs was proposed that optimizes content placement and server selection based on content popularity and geographical distribution of requests [10]. In this chapter we focus on reactive caching approaches leveraging video structure and request announcements rather than proactively placing content

into network caches. Other approaches use a two-tier cache that distinguishes between to-be-played and possibly played segments, partitioning the cache in two layers that are dynamically scaled [11]. The approach of dynamic cache partitioning has also been applied by *Wauters et al.*, where caching decisions in a time-shifted television service are based on content popularity metrics [12]. Popularity-based caching approaches have also been proposed by *De Vleeschauwer et al.* [13].

Marinca et al. use an analysis of the clients' request over passed time intervals to predict the content that will be requested in the near future [14]. By using a history window of the past 24 hours allows reducing the traffic with about 30%. Others evaluate the impact of dynamic sizing between prefix and suffix caching for segmented video [15]. *Hong et al.* propose a ranking scheme that follows the dynamicity of the video library [16]. The rank is based on the linearity of the video that if a chunk is requested, the next chunk will be requested with high probability in the near future. To this end each segment is assigned a value based on the number of viewers who are watching the video and will probably request the segment in the future. This value reflects the number of hits that this particular segment will receive in the future. Based on this value a rank is calculated that evicts the segments with the least chance of being reused in the future. In this chapter, we additionally take into account the timing information of future requests by considering video request announcements to further increase the cache performance.

The MIN cache replacement algorithm proposed by *Belady et al.* is an offline eviction strategy that has been proven to be optimal [1]. It requires advanced knowledge of the requested content, and based on this information, chooses to evict the item in the cache whose next request occurs furthest in the future. *Van Roy et al.* propose a proof of optimality for the MIN cache replacement algorithm based on a dynamic programming argument [17]. *Wu et al.* make use of the natural linear time structure of segmented video to propose a caching algorithm based on the exact reuse time [18]. This reuse time indicates when a cached segment will be reused and allows the eviction strategy to determine the segment request that will occur the furthest in the future. The authors do not take into account the delivery times between the different nodes in the network, only a single cache is assumed and the sessions are never interrupted. Our proposed approach not only uses the temporal structure of segmented video but also includes information of announced sessions. Furthermore, we also use interruption models to simulate user churn and use a hierarchy of caches where the delivery times between the different nodes are taken into account.

Recently, protocols have been proposed that support the announcement of deadlines, applied in this work. Shared Content Addressing Protocol (SCAP) is a transport protocol aimed at optimizing the delivery process that allows in-network intermediary elements to participate in delivering the content [19]. Furthermore it offers support for intermediary caches, multicast-like delivery of shared content

requests and announcing deadlines for the delivery of the content. Similar features are also provided by Information-Centric Networking (ICN) architectures [20]. These architectures allow content to be duplicated everywhere in the network, enabling in-network caching at a fine granularity [21].

4.2.2 Cache coordination strategies

In traditional hierarchical web caching systems there is no cooperation between the different nodes which deploy a replacement algorithm locally (e.g., LRU). Unsatisfied requests are forwarded up the hierarchy, resulting in copies of the requested objects being placed in every cache along the return path to the requester. This data replication leads to a poor utilization of the global cache storage capacity. *Che et al.* propose a cooperative hierarchical caching strategy where caching an object in a certain node is based on the access frequency for the document at this level [22, 23]. Furthermore, a document that is replaced at a certain level is further cached at the upper level cache, if not already. This ensures both that popular items reside closer to the end-user, while avoiding that documents with decreasing popularity are completely removed from the cache hierarchy.

Tang et al. investigated cache management strategies for en-route web caching and proposed a caching scheme that integrates both object placement and replacement policies [24]. In the proposed scheme, the cache status information along the routing path of a request is used to dynamically determine where to cache the object and which objects to replace. *Jiang et al.* extend upon this work and argue that previous solutions only solved restricted partial problems. The authors propose a dynamic programming approach which computes the global optimal solution with and without prefetching [25]. Previous work on hierarchical and en-route caching focused on web caching, which is substantially different compared to the caching of multimedia objects. In more recent work, *Poularakis et al.* focus on the delivery of VoD content in hierarchical cache topologies such as Internet Protocol television (IPTV) networks tend to have [26]. A heuristic approach is taken in which items are iteratively replaced by other items at the caching nodes if this yields a reduction in access cost, which is impacted by both latency and bandwidth consumption. The authors assume unit size objects which allows them to calculate an optimal solution to the content placement problem in polynomial time.

Also in CDN research, cooperative caching has been a subject of interest over the past years. *Ni et al.* propose a hierarchical overlay architecture for CDNs where different servers cooperate through intracluster and intercluster cooperative caching schemes [27]. For intracluster cooperation a portion of each cache is assigned to cooperate in a hashing based system, while for intercluster cooperation a query-based scheme is proposed where each cluster representative queries its neighboring clusters for missing content. The time-shifted and VoD nature of

new and emerging video services defies the broadcast paradigm of the underlying conventional TV networks. To cope with the increased bandwidth demands these services incur, *Borst et al.* formulate the content placement problem as a linear program in order to benchmark the globally optimal performance [28]. A lightweight cooperative content placement algorithm is proposed in which each nodes modifies its local content store to achieve the maximum gain in global utility. *Zhang et al.* performed a survey on cooperative caching in CDNs and conclude that increased cooperation between caches is required to decrease the bandwidth requirements and caching redundancy [29]. This is also confirmed by *Li et al.* who propose a genetic algorithm implemented on the MapReduce framework for the placement of video chunks [30]. The authors achieve similar hit ratios as with LRU, but are able to reduce the workload on the congested links.

In ICN, one of the simplest non-cooperative caching strategies is called Leave Copy Everywhere (LCE), which copies the object at each node along the downloading path. In a recent survey, different strategies for both explicit and implicit cache coordination are discussed [31]. *Li et al.* propose a cooperative caching strategy which aims at decreasing the cross-domain traffic for on-demand video streaming. They propose a hashing-based neighborhood caching strategy in which each node only stores chunks of which the hash of the id matches their assigned range [32]. Another proposed approach is to use age-based caching algorithms where replicas closer to the edge and with higher popularity are assigned with a higher age [33]. The routers adjust the contents age field as it travels along the path. The paper claims that it is straightforward to use the scheme for chunk-level caching, however no measures are proposed to take advantage of the temporal structure of these chunks. *Sourlas et al.* propose to put distributed autonomic managers in the cache-enabled nodes which decide on the information items to cache. The authors propose different levels of autonomicity, each with a different degree of communication overhead [34]. *Li et al.* propose an optimal strategy for provisioning storage capacity which optimizes the overall network performance and cost [35]. A routing-aware content placement algorithm is proposes which runs on a centralized server and assigns content to store to each Content-Centric Networking (CCN) router. There exists a trade-off between the content coordination cost (provisioning storage) and network routing performance in CCN. In non-coordinated caching, less distinct contents are stored and reduce the efficiency of the caching ”cloud”.

4.3 Client messaging behavior

In the approaches that will be presented in the next section, the caching nodes keep track of the start times of video streaming sessions. To this end, the caches rely on messaging from the video clients. The clients can send information in the

form of three types of messages: (i) session announcements, (ii) session initiation notifications and (iii) session finishing/canceling notifications. In this section, the messaging behavior of the video client is described. The client sends each message to the first cache on the path to the content server. How the messaging is handled inside the network depends on the caching algorithm and will be described in Section 4.4. To enhance the clarity of the remainder of this work, it should be emphasized that a *streaming session* is referred to as consuming a single video while a *sitting* consists of multiple consecutive streaming sessions of the same user.

4.3.1 Session announcements

As described earlier, users often watch multiple consecutive episodes of the same series in a single sitting. This phenomenon is often referred to as binge watching. Therefore, when a user watches an episode of a specific series, there is a significant probability that the user will watch the next episode of the same series once the current episode finishes. The client can use this information to generate a session announcement. When the client starts streaming episode i of a specific series at time t_i, the predicted start time of episode $i+1$ can be calculated as $t_{i+1} = t_i + d_i$, where d_i denotes the duration of episode i. The announcement for episode $i + 1$ contains the video ID of the next episode and its estimated start time t_{i+1}. This announcement message can be sent into the caching network anywhere between time t_i and t_{i+1}, based on the relative announcement delay parameter $\beta \in [0; 1]$. Concretely, the announcement will be send at time $a_{i+1} = t_i + \beta \times d_i$. It is important to note that no prior announcements are made for the first episode of a sitting. Furthermore, for episodic content, this strategy introduces exactly one false announcement per sitting (i.e., the announcement made for the episode after the current one when the current one is abandoned). For movies, no announcements are made.

4.3.2 Session initiations

When the playout of a video is initiated, the client informs the caching network about the video that is started and the actual start time of the streaming session. It is important to note that this messaging could also be performed implicitly by interpreting the first segment request of a specific video. For clarity reasons however, explicit messaging of session initiation is assumed in the remainder of this work.

4.3.3 Session expiration

At the end of a streaming session, the video client informs the caching network that the current video playout is finished. When an announcement was made earlier for the next episode of a series being watched, this announcement can optionally be canceled. The message used to finish a streaming session or to cancel an announcement contains the ID of the finished or canceled video and the (announced) start time of the video.

4.4 Session-aware cache replacement

The knowledge about start times of video streaming sessions can be used by the caching algorithms to infer the future request times of individual video segments. Based on this information, replacement decisions can be made. In this work, two different caching strategies are proposed. First, Section 4.4.1 describes an announcement-based caching strategy with a basic form of communication in the caching network, as proposed in our previous work [3]. Next, additional coordination between the caches is added in the form of an election strategy, as will be described in Section 4.4.2. For both caching strategies, the message handling process and the segment replacement strategy will be discussed.

The goal of both coordination strategies is to find a node in the caching network that will take the responsibility of caching the video for each streaming session. In the remainder of this chapter, a node taking responsibility of caching a specific video will be denoted as that node *accepting* the streaming session.

4.4.1 Threshold-based caching strategy

As described in Section 4.3, streaming session can either be announced prior to playout or be advertised at the moment the session starts. In the threshold-based caching strategy, sessions that were announced in advance play a different role in the replacement strategy than sessions that were only advertised at their start. Therefore, each caching node keeps track of two sets of streaming sessions. The set of announced sessions Ann will be maintained in such a way that it contains all announced sessions that were accepted by the local caching node. These sessions can either be active sessions or future sessions. Similarly, the set of perceived sessions Per will at any time contain all active streaming sessions that the cache is aware of. These sessions can either have been announced but not accepted locally or not have been announced at all. For Per, no future sessions are considered. For each session in Ann and Per, the streamed video v and its start time t_v is recorded.

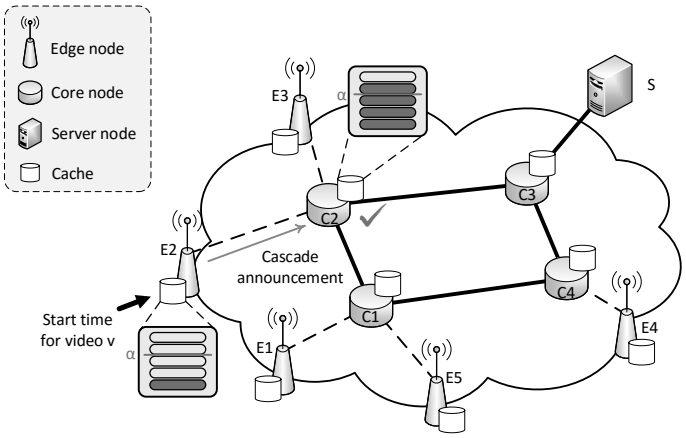


Figure 4.1: Illustration of the session announcement handling process in the threshold-based caching strategy.

4.4.1.1 Message handling

To keep track of the session information, a basic form of communication exists between the different caches in the network. This communication is implemented as a threshold-based cascading strategy.

Session announcements When a caching node receives a session announcement message, it decides autonomously whether to accept the session locally or to cascade it to the next hop on the path to the content server. A node will only decide to accept the announced session, and add it to the set of accepted announced sessions Ann, if it estimates to be able to serve at least a relative part α of the video from its local cache. This estimation is based on the number of segments of the considered video that are already present in the cache, combined with the segments that are estimated to arrive before the new session starts, based on known earlier sessions for the same video. The value of α serves as a session acceptance threshold and provides a basic form of coordination between the caches. If the node is not able to meet the threshold α, it does not accept the session and cascades the announcement to the next hop. This cascading process continues until either the announcement is accepted or the last hop on the path to the server is reached. An illustration of this process is given in Figure 4.1. A client sends a session announcement for video v to the edge node $E2$. As the number of cached segments for video v is below the session acceptance threshold α, $E2$ cascades the announcement to the next hop $C2$ where it is accepted and the cascading process is terminated.

Session initiations Upon arrival of a session initiation message for video v, the cache checks if it already has an accepted announced session for that video with the current time as start time (i.e., if Ann already contains a start time t for video v). If this is not the case, the session information is added to the set of perceived sessions Per. In this way, Ann and Per are fully distinct. However, it is important to note that even though Ann and Per are fully distinct, both sets can contain different start times t_v for the same video v. Furthermore, a single set can contain multiple start times for the same video. Next, the session initiation message is sent to the next hop on the path to the server until the last hop is reached.

Session expiration When a streaming session ends, the session information can be removed from the caching nodes. When a cache receives a session expiration message for an announced start time t of video v, it removes the session from the set of accepted announced sessions Ann. If the session was not accepted locally, the expiration message is cascaded to the next hop. Otherwise, the cascading process is finished. Perceived sessions $s_v \in Per$ are only removed when the session is known to have ended, even if it was interrupted prematurely. For a session s_v with start time t_v, this is the case when $t > t_v + L_v$, with t and L_v denoting the current time and the total length of video v respectively.

4.4.1.2 Replacement strategy

Using the information contained in the start times of the announced and perceived sessions, each node can decide which segments to keep in its cache, and which can be removed. For this purpose, the concept of earliest reuse times is applied. For each start time t_v for a specific video v, the reuse time $r_{v_x}^{t_v}$ of each segment v_x of that video can easily be calculated using (4.1), where x and l_v respectively denote the sequence number of segment v_x and the segment length for video v. If $r_{v_x}^{t_v} < t$, with t denoting the current time, the reuse time is set to infinity. The earliest announced reuse time $e_{v_x}^A$ is then obtained as the lowest reuse time $r_{v_x}^{t_v}$ of all start times t_v of accepted announced sessions Ann for video v, as expressed in (4.2). Similarly, the earliest perceived reuse time is obtained using (4.3). When there are no accepted announced or perceived sessions for video v, respectively $e_{v_x}^A$ or $e_{v_x}^P$ are said to be equal to infinity for every segment v_x of that video.

$$r_{v_x}^{t_v} = t_v + x \times l_v \tag{4.1}$$

$$e_{v_x}^A = \min_{t_v \in Ann} r_{v_x}^{t_v} \tag{4.2}$$

$$e_{v_x}^P = \min_{t_v \in Per} r_{v_x}^{t_v} \tag{4.3}$$

When a segment v_x of video v arrives at a node, the node has to decide whether or not to add it to its local cache C, if it is not already cached. If the

remaining available caching capacity is insufficient to store the new segment v_x, another segment can be selected for removal. This procedure is outlined in Algorithm 4.1. First, the earliest announced and perceived reuse times $e^A_{v_x}$ and $e^P_{v_x}$ of the newly arrived segment are calculated as described above (line 1). Next, a segment $s' \in C \cup \{v_x\}$ is selected as a candidate for eviction. This segment s' is selected as the segment with the maximal earliest announced reuse time: $s' = \arg\max_{v'_i \in C \cup \{v_x\}} e^A_{v'_i}$ (lines 4-6). It is important to keep in mind at this point that when there are no accepted sessions for the video v' of a segment v'_i, the earliest announced reuse time $e^A_{v'_i}$ is equal to infinity. In this way, when there are cached segments for which no session was accepted locally, these are always preferred for eviction. When multiple segments with the same maximal earliest announced reuse time exist, the segment with the maximal earliest perceived reuse time is selected: $s' = \arg\max_{v'_i \in C \cup \{v_x\}} e^P_{v'_i}$ (lines 7-9). The LRU order is used as a final tiebreaker (lines 10-13). When the evicted segment s' is a cached segment ($s' \in C$), it is removed from the cache and replaced by the new segment v_x (lines 18-20).

```
 1: Calculate e^A_{v_x} and e^P_{v_x}
 2: s' ← v_x
 3: for v'_i ∈ C do
 4:     Calculate e^A_{v'_i} and e^P_{v'_i}
 5:     if e^A_{v'_i} > e^A_{s'} then
 6:         s' ← v'_i
 7:     else if e^A_{v'_i} = e^A_{s'} then
 8:         if e^P_{v'_i} > e^P_{s'} then
 9:             s' ← v'_i
10:         else if e^P_{v'_i} = e^P_{s'} then
11:             Calculate LRU ranks LRU_{v'_i} and LRU_{s'}
12:             if LRU_{v'_i} > LRU_{s'} then
13:                 s' ← v'_i
14:             end if
15:         end if
16:     end if
17: end for
18: if s' ≠ v_x then
19:     Remove s' from C
20:     Add v_x to C
21: end if
```

Algorithm 4.1: Outline of the eviction strategy of the threshold-based caching approach on arrival of a new segment v_x.

The rationale behind this approach is to keep the segments in the cache that will be needed in the nearest future. However, accepted announced sessions are

always prioritized, even if another segment has an earlier perceived reuse time. This is done to ensure that the cache will not violate the intent expressed when accepting the sessions and ending the cascading chain. It is important to note that when no sessions are announced, the algorithm behaves as a purely reuse time-based replacement strategy without coordination between the caches, comparable to the approach proposed by *Wu et al.* [18], only taking into account the temporal structure of videos in the form of earliest perceived reuse times.

4.4.2 Election-based caching strategy

To further increase the performance of the caching approach, the influence of the level of additional coordination between the different caches is investigated. The goal of this election-based caching approach is to select the optimal location to cache a specific video and avoid unnecessary duplicates on the shared network path. In this caching strategy, each caching node keeps track of both a set of known sessions Kno and a set of accepted sessions Acc. In contrast to the threshold-based caching strategy, Kno will contain both active sessions and announced future sessions and Acc will contain accepted sessions for both sessions that were announced and that were only reported at the start.

4.4.2.1 Message handling

To achieve coordination in the caching network, an election-based strategy has been applied.

Session announcements When the video client sends a session announcement message to the first cache, an election process is initiated between all caches along the path to the server. When a caching node receives the announcement, the session is added to the set of known sessions Kno. Next, the cache calculates the estimated increase in number of cache hits that would be achieved when the announced video would be cached locally. This calculation takes into account the segments that are already present in the cache and the sessions that are known up to know. To clarify this calculation, assume there is an announcement at time t for streaming video v, consisting of 100 segments of 1 second, with start time $t_x > t$. After adding this session to the set of known sessions, Kno contains 3 sessions for video v, with start times t_x (the newly announced session), $t_y = t - 30$ (a streaming session that has already started) and $t_z > t$ (a future session that was announced earlier). At the moment, suppose that 10 segments of video v are present in the cache. Given the session information, one can estimate that 270 segment requests for v will arrive in the future: as $t_x > t$ and $t_z > t$, two sessions still have to start while 30 segments have already been streamed in the session with start time t_y. As the cache only contains 10 out of 100 segments at the moment, 90 segments will still have to be

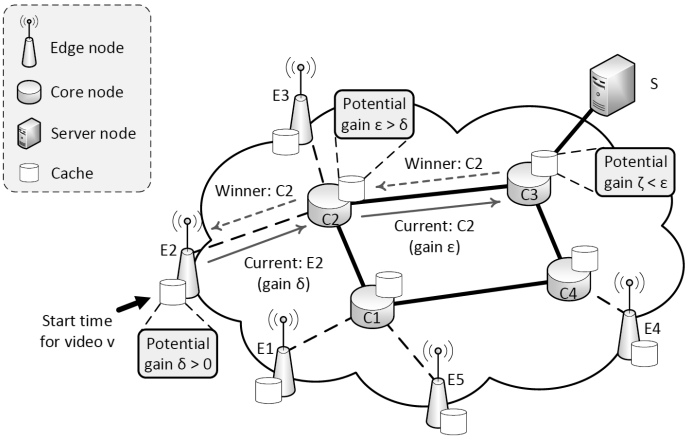


Figure 4.2: Illustration of the session announcement handling process in the election-based caching strategy.

cached, resulting in 90 cache misses. In total, we can estimate 180 future cache hits when video v would be cached.

Similarly, when there is not sufficient available capacity to store the entire video, the minimum amount of lost cache hits due to the removal of cached content is calculated. By subtracting this hit loss from the estimated additional future hits, the expected local gain can be calculated. Next, this potential local gain is passed to the next hop on the path to the server, where the same process is performed. When the last hop on the path is reached, the node with the highest potential gain is selected as the winner of the election process. When none of the caches has a positive potential gain, the election has no winner. The result of the election is cascaded back along the reverse path. When the result arrives at the winner, the announced session is added to the set of accepted sessions Acc. When segments of a video v' for which the node had accepted a session have to be removed from the cache to be able to accommodate for the newly announced session, all sessions for v' are removed from Acc. In this way, the caching capacity is guaranteed to be always sufficient to store all segments of the videos for which sessions are accepted in Acc. An illustration of the election process is given in Figure 4.2. A client sends a session announcement for video v to the edge node $E2$. The potential gain δ is calculated locally and forwarded to the next hop $C2$. There, the potential gain ϵ is calculated and forwarded to the last hop $C3$ as the gain is higher that the previous maximal gain ($\epsilon > \delta$). At node $C3$, the potential local gain is calculated as ζ, which is lower that the previous gain ϵ. As $C2$ has the highest potential gain, $C2$ is selected as the winner of the election and the result is cascaded back to the first node $E2$.

Session initiations Upon arrival of a session initiation message for video v, the caches behave similarly as for session announcements. In this case, an election is started for an implicit announcement for video v with start time t being the current time. The details of the election process are described above.

Session expiration When a streaming session ends, the session information can be removed from the caching nodes. When a cache receives a session expiration message for a streaming session of video v with start time t, the session information is removed from both the set of known sessions Kno and the set of accepted sessions Acc.

4.4.2.2 Replacement strategy

For the segment replacement strategy, a reuse time-based algorithm is applied, similar to the one presented earlier for the threshold-based strategy. However, as reuse times are calculated based on all known sessions, no distinction is made between announced or perceived reuse times. The earliest reuse time e_{v_x} of a segment v_x of video v is obtained as the lowest reuse time $r_{v_x}^{t_v}$ of all start times t_v of known sessions Kno for video v, as expressed in (4.4). When there are no known sessions for video v, e_{v_x} is said to be equal to infinity for every segment v_x of that video.

$$e_{v_x} = \min_{t_v \in Kno} r_{v_x}^{t_v} \tag{4.4}$$

The cache replacement procedure is outlined in Algorithm 4.2. First, the earliest reuse time e_{v_x} is calculated as described above (line 1). Next, a segment $s' \in C \cup \{v_x\}$ is selected as a candidate for eviction. However, segments are only considered for eviction if they do not belong to a video for which a session was accepted (line 4). The segment s' is selected as the segment with the maximal earliest reuse time: $s' = \arg\max_{v'_i \in C \cup \{v_x\}} e_{v'_i}$ (lines 5-7). When multiple such segments exist, the LRU order is used as a tiebreaker (lines 8-11). When the evicted segment s' is a cached segment ($s' \in C$), it is removed from the cache and replaced by the new segment v_x (lines 16-18).

The rationale behind this approach is again to keep the segments in the cache that will be needed in the nearest future. However, by not considering segments for removal when a session for that video was accepted, we ensure that the cache will always keep the segments that it intended to store when the session was accepted. In contrast to the threshold-based caching strategy, the election-based strategy provides coordination between the caches, even when no sessions are announced.

```
Calculate e_{v_x}
s' ← v_x
for v'_i ∈ C do
    if ∄ start time t_{v'} ∈ Acc for video v' then
        Calculate e_{v'_i}
        if e_{v'_i} > e_{s'} then
            s' ← v'_i
        else if e_{v'_i} = e_{s'} then
            Calculate LRU ranks LRU_{v'_i} and LRU_{s'}
            if LRU_{v'_i} > LRU_{s'} then
                s' ← v'_i
            end if
        end if
    end if
end for
if s' ≠ v_x then
    Remove s' from C
    Add v_x to C
end if
```

Algorithm 4.2: Outline of the eviction strategy of the election-based caching approach on arrival of a new segment v_x.

4.5 Scenario description

To evaluate the proposed approaches, a VoD request trace has been constructed based both on a request trace of the VoD service of a leading European telecom operator and on statistics provided by Conviva[2]. In October 2015, Conviva opened access to viewer experience data, based on the analysis of 4 billion video streams per month spread across 180 countries[3]. The dataset contains statistics about the last year, ranging from Q4 2014 to Q3 2015. Section 4.5.1 describes the main characteristics of the constructed request trace, while the applied session duration model, simulating the session interruption behavior, is described in Section 4.5.2. Finally, the considered network topologies are described in Section 4.5.3.

4.5.1 VoD trace characteristics

4.5.1.1 Content type

A wide variety of streaming services is monitored by Conviva, with video content consisting of 4 categories: live videos, short videos ($\leq$ 15 minutes), episodic content and movies. As this chapter focuses on a VoD scenario, only episodic content and movies are considered. In the remainder of this chapter, we will use the terms

[2]Conviva - http://www.conviva.com

[3]Conviva dataset - http://www.conviva.com/industry-data-portal/

episodic content and series interchangeably. According to the Conviva dataset, over the last year, 82.28% of the monitored streaming sessions was for episodic content while 17.72% was for movies.

However, while the Conviva dataset contains information about the streaming sessions, it has no statistics about the content catalog. To identify a realistic ratio between series and movies in a VoD content catalog, the Netflix catalog in the USA has been analyzed[4]. At the time of the analysis, the USA Netflix catalog contained about 7720 movies and 2920 seasons of series.

4.5.1.2 Content characteristics

According to the Conviva dataset, the average bit rate amounts to 2.53Mbps and 1.92Mbps for movies and episodic content respectively. As no information about the catalog is available, the average video duration and the number of episodes per series season were set based on natural values. The duration of a movie was set to be between 60min and 150min while an average series episode has a duration of 20min to 50min. All videos are considered to have a segment duration of 1s. Each season of a series consists of 10 to 30 episodes and the popularity of a series is uniformly distributed across the episodes.

4.5.1.3 Binge watching behavior

In August 2015, Conviva released the results of a survey on binge watching [2]. Figure 4.3 shows the distribution of binge watching durations as published in this report. It can be seen that on average, 2.32 consecutive episodes of a series are watched in a single sitting. The binge watching session can start with any episode of the series.

4.5.1.4 Global content popularity

Given the long-tailed nature of content popularity, the global popularity distribution is commonly represented by the Zipf law [36]. When the content catalog consists of N videos and the videos are ranked according to decreasing global popularity, the relative number of requests r_n for the video with rank n is calculated according to (4.5), where μ is the scaling-law coefficient of the Zipf distribution. Based on an analysis of a request trace of the VoD service of a leading European telecom operator, in this work, the value of μ was set to 0.9 for both movies and episodic content.

$$r_n = \frac{n^{-\mu}}{\sum_{k=1}^{N} k^{-\mu}} \tag{4.5}$$

[4] iStreamguide - http://usa.istreamguide.com

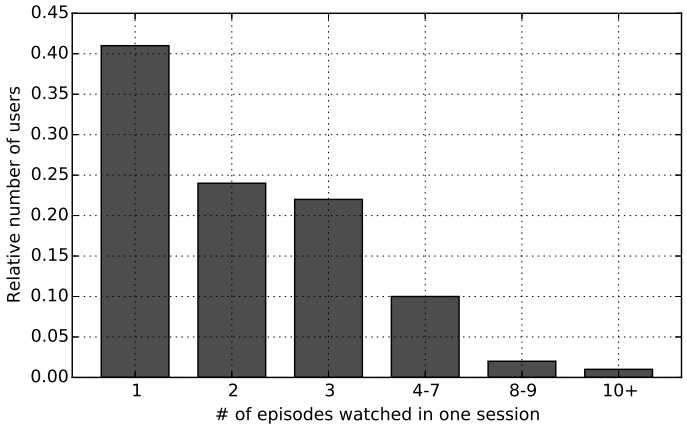


Figure 4.3: Distribution of the duration of binge watching sessions as reported by Conviva [2].

4.5.1.5 Geographical distribution of requests

While the Zipf law models the global popularity distribution, it does not account for the geographical distribution of requests. In this work, the requests are considered to be distributed geographically according to the model proposed by *Tuncer et al.* [37]. When the content catalog consists of N videos and requests originate from L locations, the video with rank n is requested from l_n locations, as defined in (4.6). In this equation, σ is a strictly positive parameter that defines the characteristics of the distribution. Again based on an analysis of a request trace of the VoD service of a leading European telecom operator, the value of σ was identified to be 2.43.

$$l_n = 1 + (L-1) \times \left(\frac{N-n}{N-1} \right)^{\sigma} \tag{4.6}$$

4.5.1.6 Request pattern

In order to distribute the video requests over time, we analyzed a request trace of the VoD service of a leading European telecom operator, collected between Saturday February 6, 2010 and Sunday March 7, 2010. Based on this analysis, the weekly request pattern shown in Figure 4.4 has been extracted. This graph shows the distribution of requests during the week, starting in Monday 00h00min (day 0) up to Sunday 23h59min (day 7), and the average daily trend.

Using the above characteristics, a VoD request trace of 7 days has been generated, consisting of 125,000 requests spread across 12 locations. The main characteristics of the resulting VoD request trace are summarized in Table 4.1. The total content catalog size amounts to about 4.25Tbyte.

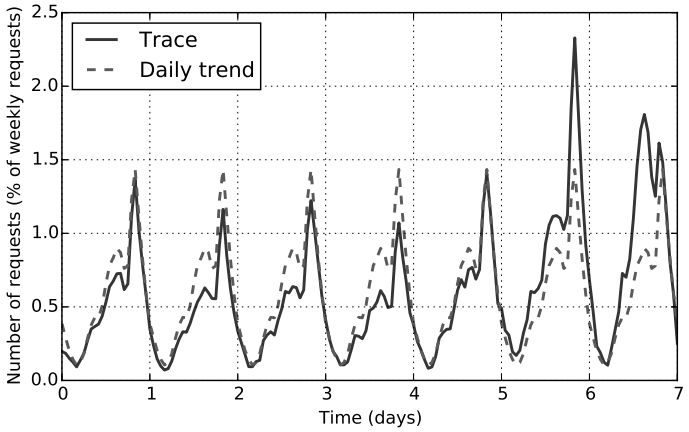


Figure 4.4: Relative distribution of the weekly requests starting from Monday 00h00min (day 0) up to Sunday 23h59min (day 7).

Table 4.1: Summary of resulting VoD trace characteristics.

Characteristic	**Movies**	**Series**
Units	700	265
Episodes/unit	1	10-30
Rel. nr. of requests	17.72%	82.28%
Avg. bit rate	2.53Mbps	1.92Mbps
Duration	60-150min	20-50min
Avg. binge watching duration	1	2.32
Popularity distribution	Zipf($\mu = 0.9$)	Zipf($\mu = 0.9$)
Geographical distribution	$l_n(\sigma = 2.43)$	$l_n(\sigma = 2.43)$

4.5.2 Session duration

In contrast to most other related work, we take into account the fact that users do not necessarily stream a video entirely, but may interrupt the session mid-stream. Multiple models have been presented in literature to represent the session duration in video streaming services. *Ullah et al.* provide a survey of user behavior measurements in several types of video streaming services [38]. According to this survey, the session duration of most VoD services can be modeled using an exponential distribution. To deal with the fixed content length in this work, a normalized exponential distribution was used to model the session duration in our experiments. The cumulative probability $p(x)$ of a session to last at most a relative part $x \in [0; 1]$ of the video is calculated as shown in (4.7), where the value of λ depends on the video type.

$$p(x) = \frac{1 - e^{-\lambda x}}{1 - e^{-\lambda}} \tag{4.7}$$

For movies, this session duration model is applied to every streaming session. On the contrary, given the binge watching behavior for episodic content, the duration model is only applied to the last episode of a binge watching session. The previous episodes in the session are considered to be watched completely. According to Conviva, the average session duration amounts 52.23% and 68.43% of the total video length for movies and series respectively. Taking into account the binge watching characteristics, only 26.75% of the last episode of a binge watching session is watched on average. This results in the parameter of the normalized exponential duration model set to $\lambda = -0.21$ for movies and to $\lambda = 3.34$ for series.

4.5.3 Network topology

To evaluate the performance of the proposed algorithm in a distributed scenario, two network topologies with different characteristics have been used. In both topologies, a distinction is made between core nodes and edge nodes. All requests are introduced at the edge nodes, while the core nodes interconnect the entire network. Caching capacity is available both at the core nodes and edge nodes. As described in Section 4.5.1, the employed VoD request trace contains 12 cities, which we map onto 12 edge nodes in both topologies.

The first topology represents a three-level tree topology, consisting of 12 edge nodes, 5 core nodes and 1 server node S, hosting all video content. The caching capacity at core nodes $C2$-$C5$ is twice as high as the capacity at the edge nodes, while the capacity of the root core node $C1$ is twice as high as the capacity of the other core nodes. The resulting topology is shown in Figure 4.5.

The second topology represents a general network, based on the GÉANT topol-

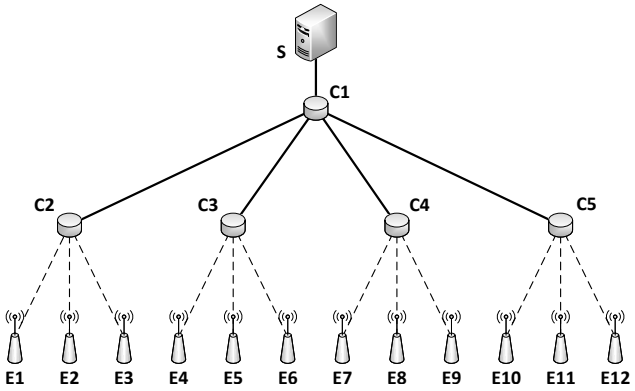


Figure 4.5: Evaluated tree topology.

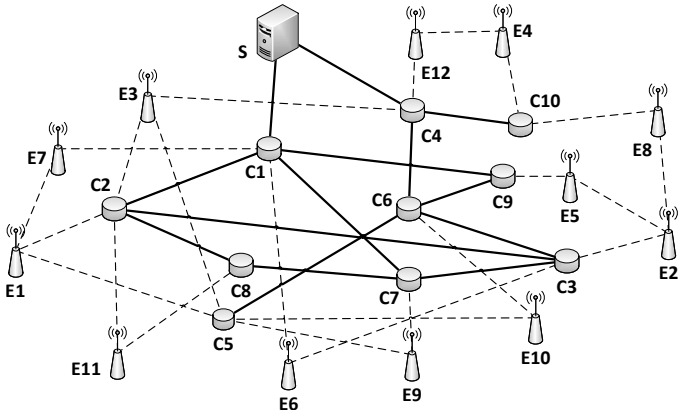


Figure 4.6: Evaluated GÉANT-based topology.

ogy[5], consisting of 12 edge nodes, 10 core nodes and 1 server node S, hosting all video content. The caching capacity at the core nodes is twice as high as the capacity at the edge nodes. The resulting topology is shown in Figure 4.6.

In each experiment, the total caching capacity in the network is expressed relative to the total video catalog size. The capacity is spread uniformly across the network, taking into account the capacity ratios described above.

[5]GÉANT Project - http://www.geant.net

4.6 Evaluation results

To characterize the performance of the proposed approaches, multiple performance indicators are evaluated:

- **Hit ratio:** the relative amount of segment requests that could be served from within the Internet Service Provider (ISP) network (in %).
- **Average bandwidth usage:** the average bandwidth usage in the entire ISP network (in Mbit/s), introduced by the video streaming sessions.
- **Average hop count:** the average number of links a segment crosses between its storage location and the requesting client. This metric indicates how close the relevant content is stored to the end-users.

In all of the evaluations, the performance of the proposed approaches is compared to both the LRU caching strategy and a purely reuse time-based strategy without session announcements and cache coordination, comparable to the approach proposed by *Wu et al.* [18].

4.6.1 Influence of the session acceptance threshold α

As described in Section 4.4.1, the session acceptance threshold α is used in the threshold-based caching approach to decide whether to accept an announced session locally or to cascade it to the next hop. To evaluate the impact of this parameter, the scenario where all announcements are made without delay (i.e., $\beta = 0.0$) is considered first.

Figure 4.7 shows the influence of the session acceptance threshold α on the hit ratio of the proposed threshold-based caching strategy for various amounts of caching capacity in the tree topology. It can be seen that the best results can be achieved with a session acceptance threshold α of 25%. Similar results are obtained for the general topology and for the other performance indicators.

However, it can be seen that for smaller amounts of caching capacity, the performance of the approach with acceptance threshold $\alpha = 0\%$ comes closer to the performance with acceptance threshold $\alpha = 25\%$. This behavior can be explained by the semantics of the acceptance threshold. With an acceptance threshold of 0%, all sessions are accepted at the edge nodes, without cascading information in the network. This clearly is a suboptimal situation. However, when for example the acceptance threshold amounts 25%, sessions will only be accepted when at least 25% of the segments of that session are expected to be present in the cache. For small caches, chances strongly decrease that a significant amount of segments will be cached at once when multiple sessions are active in parallel. Therefore,

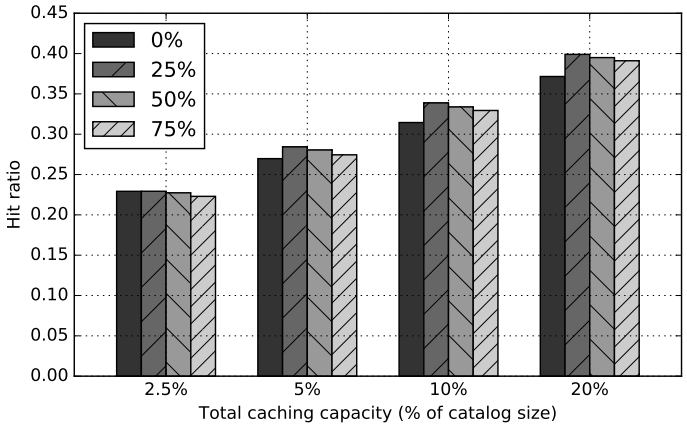


Figure 4.7: Impact of the session acceptance threshold α on the hit ratio of the threshold-based caching strategy in the tree topology when all announcements are made without delay.

only very few sessions will be accepted inside the network and most of the announcement information will be unused. In this case, better results are achieved by accepting all sessions at the network edge, rather than not accepting them at all.

The impact of the threshold α on the session acceptance in the network is shown in Figure 4.8 for a caching capacity of 5% of the total content catalog in the tree topology. The results for $\alpha = 0\%$ are omitted for visibility reasons, as in this case all sessions are accepted at the edge nodes. As expected, the number of accepted sessions decreases with higher values of α. Furthermore, when α increases, a bigger part of the accepted sessions was accepted in the network core nodes. This can be explained by the fact that core nodes simultaneously perceive requests of multiple edge nodes, such that cached segments can be reused across geographical locations. Similar results are obtained for the general topology.

Given the above analysis, the session acceptance threshold of $\alpha = 25\%$ can generally be selected as the best configuration for the threshold-based caching strategy.

4.6.2 Influence of the relative announcement delay β

As introduced in Section 4.3, the relative announcement delay parameter β defines at what time in a streaming session of a series episode, the next episode will be announced. Naturally, the value of β serves as a tradeoff. When announcements are made in an early stage (low values of β), the caching network is given a larger future request window, based on which more accurate decisions can be made. On the other hand, when delaying the announcements (larger values of

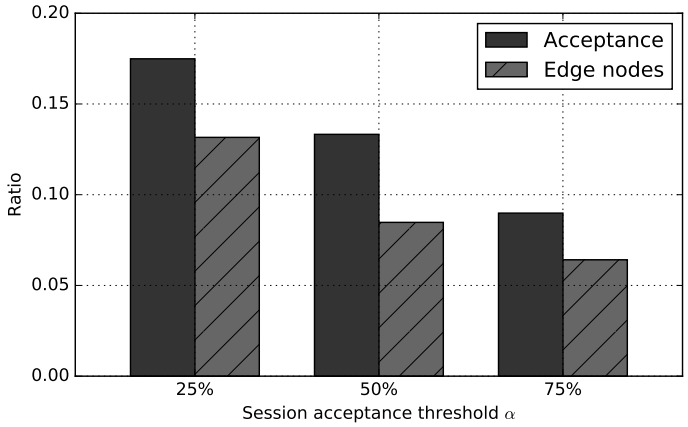


Figure 4.8: Impact of the session acceptance threshold α on the session acceptance in the network for the threshold-based caching strategy in the tree topology when all announcements are made without delay and with a total capacity of 5% of the catalog size.

β), false announcements are avoided given that some streaming sessions will have ended before an announcement was made.

The generated VoD trace contains 44091 sittings for series, in total streaming 102627 episodes. In theory, when all announcements are made immediately, 44091 false announcements will be made when announcing the next episode during the last episode of the sitting (neglecting the situation when the last episode of a season is being watched and no future announcement can be made). Figure 4.9 shows the number of false announcements, relative to the maximal number of false announcements, for the relative announcement delay $\beta \in \{0.0, 0.25, 0.50, 0.75\}$. As expected, the number of false announcements decreases with higher values of β. Furthermore, the number of false announcements clearly follows the theoretical trend that can be calculated based on the session duration model. More specifically, the theoretical number of false announcements for a given value of β is the number of streaming sessions that have not finished after a relative part β of its total duration.

The influence of the relative announcement delay parameter β on the performance of the proposed caching strategies in terms of hit ratio is shown in Figure 4.10, using the tree topology and a total caching capacity of 5% of the total catalog size. It can be observed that while the number of false announcements significantly decreases with increasing values of β, better performance is achieved for lower values of β for both the threshold-based and the election-based caching strategies. As shown in Table 4.2, the same influence can be perceived on the average hop count and the average total bandwidth usage. Similar results are ob-

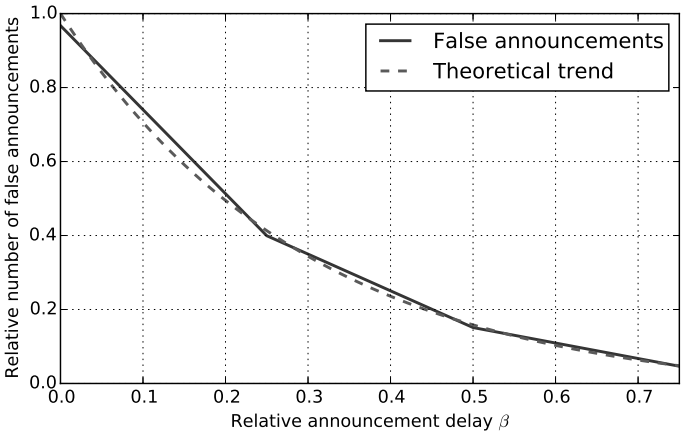


Figure 4.9: Impact of the relative announcement delay β on the relative number of false announcements and its theoretical trend.

Table 4.2: Impact of the relative announcement delay β on the average hop count and the average total bandwidth usage in the tree topology with a total caching capacity of 5% of the catalog size.

	Threshold-based		**Election-based**	
β	**Hop count**	**Bandwidth**	**Hop count**	**Bandwidth**
0	2.48	2057Mbps	2.41	2002Mbps
0.25	2.48	2061Mbps	2.43	2016Mbps
0.5	2.49	2068Mbps	2.45	2034Mbps
0.75	2.51	2077Mbps	2.48	2055Mbps

tained in the general topology and for different cache sizes. As the LRU and the reuse time-based strategies do not take into account session announcements, their performance is not influenced by the value of β.

Based on the above analysis, it can be concluded that in the considered scenario, the benefits of the earlier availability of future request information exceed the negative impact of the false announcements for both of the proposed caching strategies. Therefore, the clients are configured to immediately announce an episode of a series once the previous episode is started, i.e., $\beta = 0.0$.

4.6.3 Distribution of accepted sessions

The cooperation strategy within the caching approach defines how sessions are accepted in the caching network. Figure 4.11 shows the relative amount of sessions that were accepted in the tree topology for different amounts of caching capacity. As in the threshold-based approach only announced session can be accepted, the

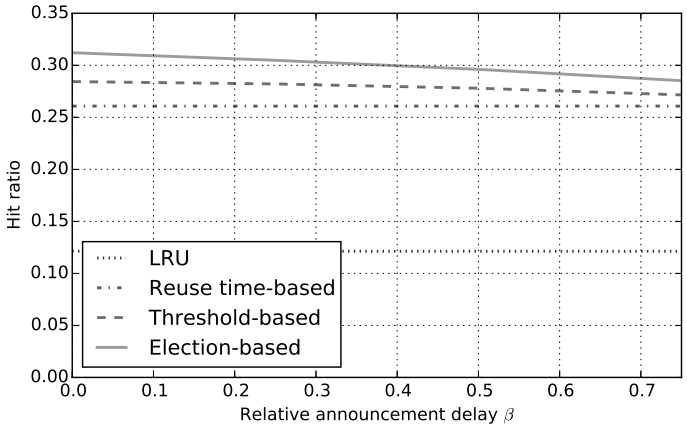


Figure 4.10: Impact of the relative announcement delay β on the hit ratio of the proposed caching strategies in the tree topology with a total caching capacity of 5% of the catalog size.

acceptance ratio is calculated based on the number of announcements. In contrast, in the election-based strategy, sessions can be accepted, independent of whether or not they were announced. Therefore, the acceptance ratio is based on the number of announcements and the number of unannounced sessions. As expected, an increasing trend can be observed with the amount of caching capacity for both approaches. However, for the election-based strategy, significantly more sessions are accepted. This difference can be explained by the nature of the communication in both approaches. For the threshold based approach, sessions will only be accepted when at least a local hit ratio of 25% can be expected. In contrast, in the election-based strategy, sessions can be accepted as soon as a hit can be expected for at least 1 segment of the video.

The cooperation strategy not only influences the session acceptance ratio, but also impacts the distribution of the accepted session across the network. Figure 4.12 shows the distribution of the accepted sessions for the threshold-based caching strategy across the edge nodes (nodes $E1 - E12$), the core nodes $C2 - C5$ and the root node $C1$ of the tree topology. It can be seen that for small amounts of caching capacity, the majority of the accepted sessions are accepted deep in the tree network. Given the threshold of $\alpha = 25\%$, most announcements are cascaded in the edge nodes. In the core nodes, however, caching space can be shared between sessions streaming the same content from different geographical locations. In this way, more sessions can reach the acceptance threshold in the core nodes and at the root node. When the capacity increases, more sessions are accepted directly at the edge nodes as the number of segments from a single video that can be cached grows. Relatively less announcements are cascaded all the way to the root

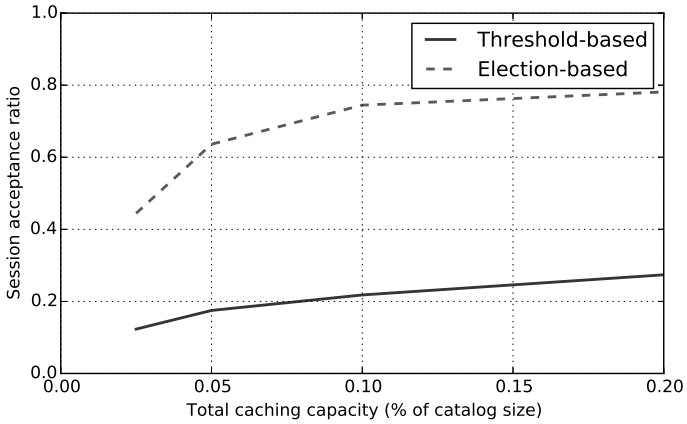


Figure 4.11: Session acceptance ratio for both of the proposed caching strategies in the tree topology for different amounts of caching capacity.

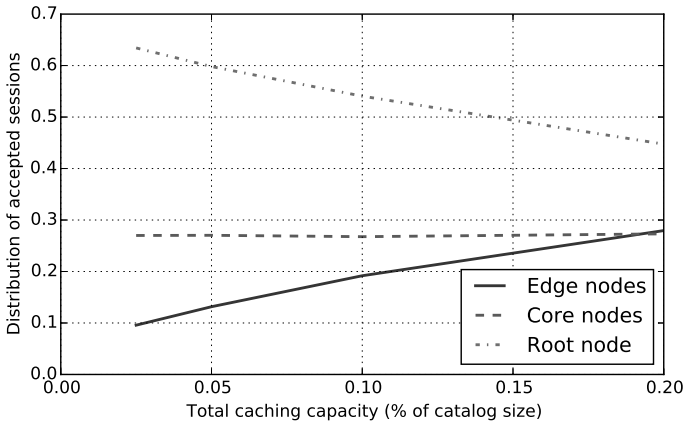


Figure 4.12: Distribution of accepted sessions in the tree topology for the threshold-based caching strategy.

of the tree network, decreasing the relative amount of sessions that are accepted there. However, most of the accepted sessions are still accepted at the core nodes or at the root of the tree.

The distribution of accepted sessions in the tree network for the election-based caching strategy is shown in Figure 4.13. Compared to the threshold-based strategy, accepted sessions are more evenly distributed across the network for small amounts of caching capacity. When the caching capacity increases, more locally popular content can be cached in the edge nodes, increasing the relative amount of accepted sessions at the edge nodes. However, when the caching capacity in-

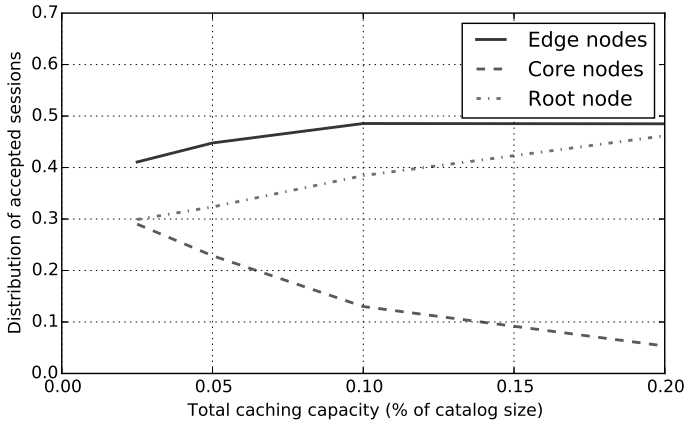


Figure 4.13: Distribution of accepted sessions in the tree topology for the election-based caching strategy.

creases, relatively more sessions are accepted in the root of the tree network as well. All sessions cross the root $C1$ of the tree network, which thus has knowledge about all active requests in the network. Caching globally popular content in the root of the network can increase local hits for sessions originating from multiple locations. Therefore, the expected gain for globally popular content is the highest in the root node. As the capacity increases, the estimated hit loss decreases, resulting in more sessions accepted in the root node.

4.6.4 Performance comparison

Finally, the performance of the proposed approaches can be compared to the LRU approach and a purely reuse time-based approach. For the threshold-based approach, based on the analysis in Section 4.6.1, the session acceptance threshold was set to $\alpha = 25\%$. The client messaging behavior was configured to immediately send out announcements for subsequent episodes ($\beta = 0.0$), based on the results of Section 4.6.2.

4.6.4.1 Tree topology

Figure 4.14 provides a comparison in terms of hit ratio between the proposed approaches and the reference approaches in the tree topology. It can be seen that both proposed approaches significantly outperform the state-of-the-art for each of the evaluated caching capacities. However, the performance gain compared to the reuse time-based approach increases for larger capacities. When more caching capacity is available, the proposed strategies can benefit more from the available

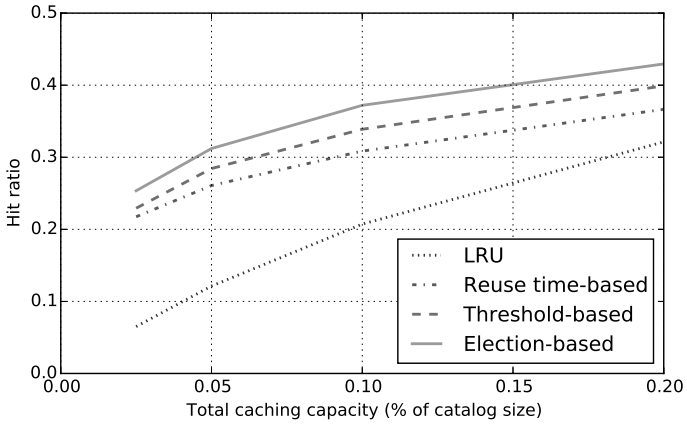


Figure 4.14: Performance comparison in terms of hit ratio in the tree topology.

session announcements. However, it can be seen that the performance increase compared to the LRU approach decreases for larger capacities, even for the purely reuse time-based approach. This demonstrates that the advantage of taking into account the temporal structure in the video is stronger for smaller caches. Comparing the threshold-based approach to the election-based approach shows that the advanced level of coordination in the caching network yields a constant performance increase, independent of the leased capacity.

The performance in terms of hit ratio, average hop count and average total bandwidth usage inside the tree network is summarized in Table 4.3. It can be seen that when the caching capacity amounts to 5% of the content catalog, using the election-based coordination can relatively increase the hit ratio with 10% while using 3% less bandwidth by bringing the content 3% closer to the end-user on average, compared to the threshold-based coordination. The proposed approach results in a hit ratio increase of 20% compared to the state-of-the-art while using 4% less bandwidth. Compared to the LRU approach, the performance increase amounts to 157%, 14% and 14% in terms of hit ratio, average hop count and average total bandwidth usage respectively.

4.6.4.2 General topology

As can be seen in Figure 4.15, similar results can be obtained in the general GÉANT-based topology as in the tree topology, showing the general applicability of our proposed approach. However, it can be seen that each of the evaluated approaches yield better performance in the tree topology. This can be explained by the hierarchical structure in the tree topology. As each node in the tree has an aggregated view of the streaming sessions in its child nodes, the same caching ca-

Table 4.3: Performance comparison in the tree topology.

	Approach	Hit ratio (%)	Hop count	Bandwidth (Mbps)
2.5%	LRU	6.51	2.89	2415
	Reuse time-based	21.74	2.61	2161
	Threshold-based	22.93	2.58	2145
	Election-based	25.35	2.52	2092
5%	LRU	12.13	2.79	2335
	Reuse time-based	26.08	2.53	2092
	Threshold-based	28.44	2.48	2057
	Election-based	31.21	2.41	2002
10%	LRU	20.72	2.62	2197
	Reuse time-based	30.85	2.43	2007
	Threshold-based	33.88	2.36	1957
	Election-based	37.20	2.30	1905
20%	LRU	32.13	2.36	1981
	Reuse time-based	36.65	2.28	1887
	Threshold-based	39.90	2.21	1827
	Election-based	42.95	2.16	1790

pacity can be used more efficiently compared to the general topology, lacking any hierarchical structure. Furthermore, the fact that the caching capacity is distributed across a higher number of nodes in the GÉANT-based topology compared to the tree topology results in lower absolute cache sizes, negatively influencing the hit ratio for all of the considered approaches.

Table 4.4 shows the performance in terms of the different performance metric for the general topology. It can be seen that in a scenario where the total caching capacity amounts to 5% of the content catalog, the election-based approach outperforms the threshold-based approach with 13%, 2% and 2% in terms of hit ratio, average hop count and average total bandwidth usage respectively. Compared to the state-of-the-art, the proposed approach results in a hit ratio increase of 22% while using 3% less bandwidth by bringing the content 4% closer to the end-user on average.

4.7 Conclusions

In this chapter, two cooperative announcement-based caching strategies for Video-on-Demand (VoD) systems were presented. These strategies not only take into account the temporal structure of videos, but also take advantage of the phenomenon of binge watching where users watch multiple consecutive episodes of the same TV show. This allows a significant part of the streaming sessions to be announced

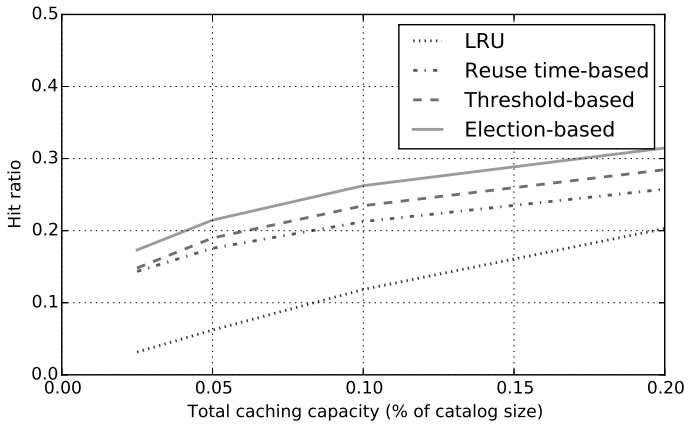


Figure 4.15: Performance comparison in terms of hit ratio in the GÉANT-based topology.

Table 4.4: Performance comparison in the general topology.

	Approach	**Hit ratio (%)**	**Hop count**	**Bandwidth (Mbps)**
2.5%	LRU	3.16	2.70	2251
	Reuse time-based	14.35	2.49	2067
	Threshold-based	14.82	2.48	2060
	Election-based	17.33	2.42	2016
5%	LRU	6.21	2.64	2204
	Reuse time-based	17.55	2.44	2017
	Threshold-based	18.99	2.41	1996
	Election-based	21.46	2.35	1953
10%	LRU	11.84	2.53	2111
	Reuse time-based	21.26	2.36	1952
	Threshold-based	23.47	2.31	1919
	Election-based	26.24	2.26	1879
20%	LRU	20.29	2.34	1956
	Reuse time-based	25.76	2.25	1862
	Threshold-based	28.46	2.19	1817
	Election-based	31.47	2.15	1784

in advance of the actual playout. While both of the proposed approaches apply a reuse time-based replacement strategy, they fundamentally differ in the coordination process inside the caching network. Both approaches have been thoroughly evaluated using a VoD request trace in two network topologies with different characteristics and with different amounts of caching capacity, showing the general applicability of the proposed approach. It was shown that the proposed approach, applying an election-based coordination strategy, can outperform the state-of-the-art with more than 20% in terms of cache hit ratio in a realistic scenario, while simultaneously using 3% less network bandwidth.

Acknowledgment

M. Claeys and N. Bouten are funded by a grant of the Agency for Innovation by Science and Technology in Flanders (IWT). The research was performed partially within the ICON SHIFT-TV project (under grant agreement no. 140684). This work was partly funded by Flamingo, a Network of Excellence project (318488) supported by the European Commission under its Seventh Framework Programme.

4.8 Addendum: GÉANT-based topology graphs

As the influence of the different parameters, evaluated in Sections 4.6.1-4.6.3, is comparable in both the tree topology and the GÉANT-based topology, graphs were only presented for the tree topology. For sake of completeness, the graphs for the GÉANT-based topology are presented below in this addendum. The considered scenarios were identical for both topologies.

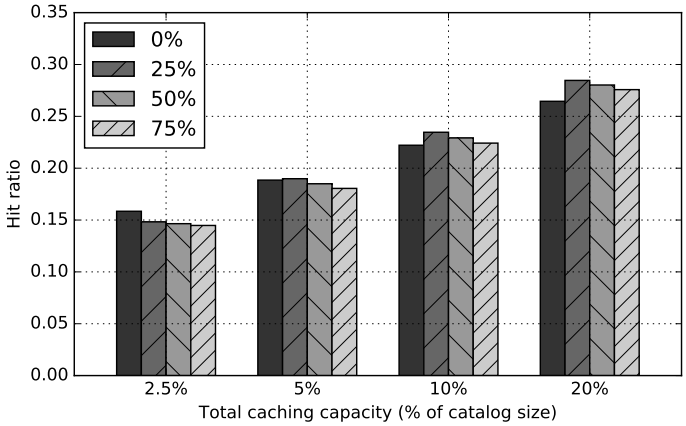


Figure 4.16: Impact of the session acceptance threshold α on the hit ratio of the threshold-based caching strategy in the GÉANT-based topology when all announcements are made without delay.

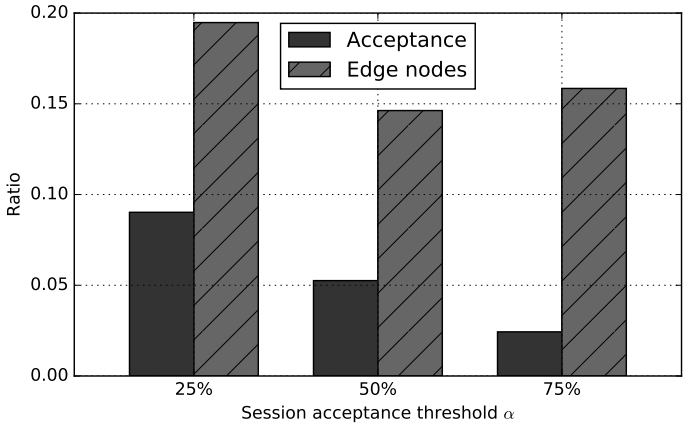


Figure 4.17: Impact of the session acceptance threshold α on the session acceptance in the network for the threshold-based caching strategy in the GÉANT-based topology when all announcements are made without delay and with a total capacity of 5% of the catalog size.

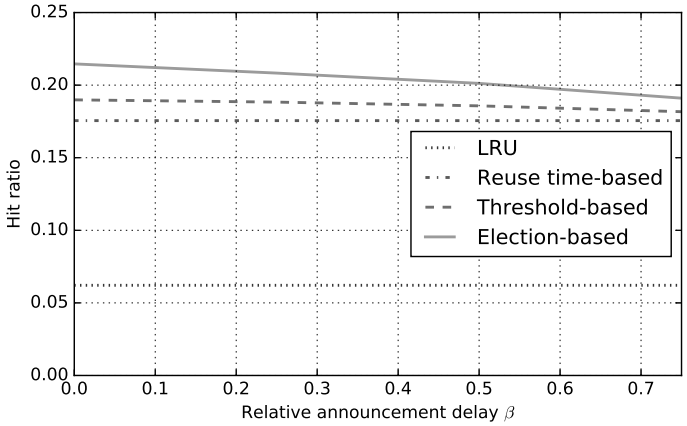


Figure 4.18: Impact of the relative announcement delay β on the hit ratio of the proposed caching strategies in the GÉANT-based topology with a total caching capacity of 5% of the catalog size.

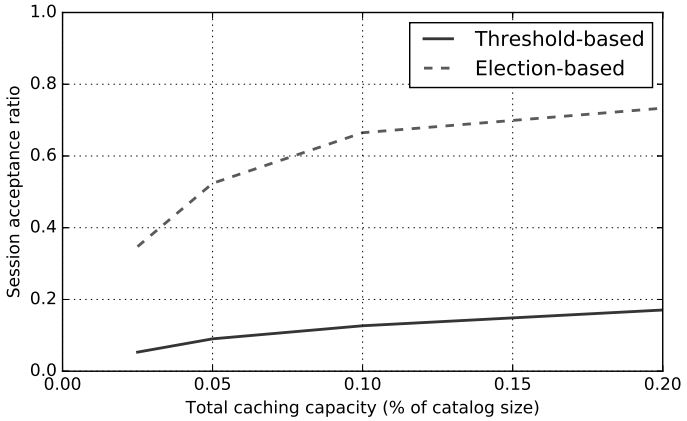


Figure 4.19: Session acceptance ratio for both of the proposed caching strategies in the GÉANT-based topology for different amounts of caching capacity.

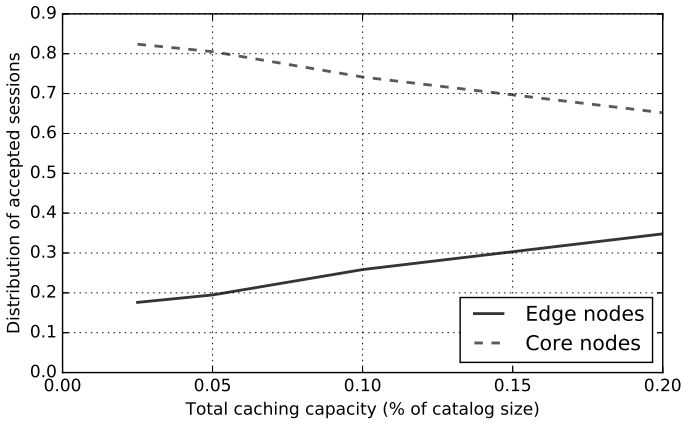


Figure 4.20: Distribution of accepted sessions in the GÉANT-based topology for the threshold-based caching strategy.

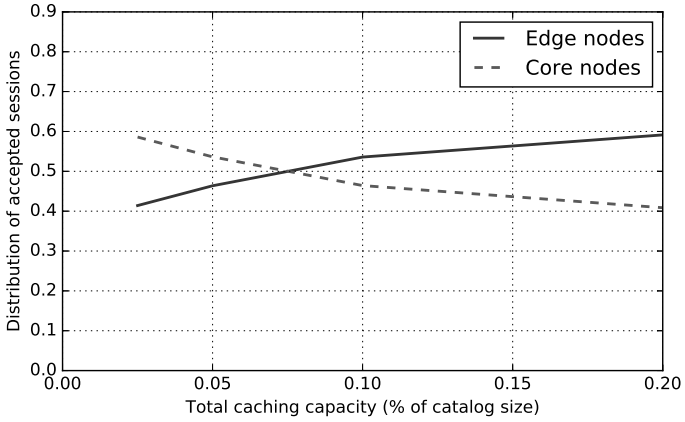


Figure 4.21: Distribution of accepted sessions in the GÉANT-based topology for the election-based caching strategy.

References

[1] L. Bélády. *A study of replacement algorithms for a virtual-storage computer*. IBM Systems Journal, 5(2):78–101, 1966.

[2] Conviva. *Binge watching: the new currency of video economics*. Technical report, Conviva, 2015.

[3] M. Claeys, N. Bouten, D. De Vleeschauwer, W. Van Leekwijck, S. Latré, and F. De Turck. *An announcement-based caching approach for video-on-demand streaming*. In Proceedings of the International Conference on Network and Service Management (CNSM), pages 1–8, 2015.

[4] N. Megiddo and D. S. Modha. *Outperforming LRU with an adaptive replacement cache algorithm*. Computer, 37(4):58–65, 2004.

[5] N. Färber, S. Döhla, and J. Issing. *Adaptive progressive download based on the MPEG-4 file format*. Journal of Zhejiang University SCIENCE A, 7(1):106–111, 2006.

[6] K.-L. Wu, P. S. Yu, and J. L. Wolf. *Segment-based proxy caching of multimedia streams*. In Proceedings of the International Conference on World Wide Web, pages 36–44, 2001.

[7] K.-L. Wu, P. S. Yu, and J. L. Wolf. *Segmentation of multimedia streams for proxy caching*. IEEE Transactions on Multimedia, 6(5):770–780, 2004.

[8] S. Chen, B. Shen, S. Wee, and X. Zhang. *Segment-based streaming media proxy: modeling and optimization*. IEEE Transactions on Multimedia, 8(2):243–256, 2006.

[9] S. Chen, B. Shen, S. Wee, and X. Zhang. *SProxy: A caching infrastructure to support internet streaming*. IEEE Transactions on Multimedia, 9(5):1062–1072, 2007.

[10] M. Claeys, D. Tuncer, J. Famaey, M. Charalambides, S. Latré, G. Pavlou, and F. De Turck. *Proactive multi-tenant cache management for virtualized ISP networks*. In Proceedings of the International Conference on Network and Service Management (CNSM), pages 82–90, 2014.

[11] K.-C. Liang and H.-F. Yu. *Adjustable two-tier cache for IPTV based on segmented streaming*. International Journal of Digital Multimedia Broadcasting, 2012(1):1–8, 2012.

[12] T. Wauters, W. Van de Meerssche, F. De Turck, B. Dhoedt, P. Demeester, T. Van Caenegem, and E. Six. *Co-operative proxy caching algorithms for*

time-shifted IPTV services. In Proceedings of the EUROMICRO Conference on Software Engineering and Advanced Applications, pages 379–386, 2006.

[13] D. De Vleeschauwer and K. Laevens. *Performance of caching algorithms for IPTV on-demand services*. IEEE Transactions on Broadcasting, 55(2):491–501, 2009.

[14] D. Marinca, A. Hamieh, D. Barth, K. Khawam, D. De Vleeschauwer, and Y. Lelouedec. *Cache management using temporal pattern based solicitation for content delivery*. In Proceedings of the International Conference on Telecommunications (ICT), pages 1–6, 2012.

[15] Z. Xu, X. Guo, Y. Pang, and Z. Wang. *The transmitted strategy of proxy cache based on segmented video*. In Proceedings of the IFIP International Conference on Network and Parallel Computing (NPC), pages 502–507, 2004.

[16] D. Hong, D. De Vleeschauwer, and F. Baccelli. *A chunk-based caching algorithm for streaming video*. In Proceedings of the Workshop on Network Control and Optimization, pages 1–8, 2010.

[17] B. Van Roy. *A short proof of optimality for the MIN cache replacement algorithm*. Information Processing Letters, 102(2):72–73, 2007.

[18] T. Wu, K. De Schepper, W. Van Leekwijck, and D. De Vleeschauwer. *Reuse time based caching policy for video streaming*. In Proceedings of the IEEE Consumer Communications and Networking Conference (CCNC), pages 89–93, 2012.

[19] K. De Schepper, B. De Vleeschauwer, C. Hawinkel, W. Van Leekwijck, J. Famaey, W. Van de Meerssche, and F. De Turck. *Shared Content Addressing Protocol (SCAP): Optimizing multimedia content distribution at the transport layer*. In Proceedings of the IEEE Network Operations and Management Symposium (NOMS), pages 302–310, 2012.

[20] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. *A survey of information-centric networking*. IEEE Communications Magazine, 50(7):26–36, 2012.

[21] I. Psaras, W. K. Chai, and G. Pavlou. *Probabilistic in-network caching for information-centric networks*. In Proceedings of the ICN workshop on information-centric networki, pages 55–60, 2012.

[22] H. Che, Z. Wang, and Y. Tung. *Analysis and design of hierarchical web caching systems*. In Proceedings of the IEEE International Conference on Computer Communications (INFOCOM), pages 1416–1424, 2001.

[23] H. Che, Y. Tung, and Z. Wang. *Hierarchical web caching systems: modeling, design and experimental results*. IEEE Journal on Selected Areas in Communications (JSAC), 20(7):1305–1314, 2002.

[24] X. Tang and S. T. Chanson. *Coordinated en-route web caching*. IEEE Transactions on Computers, 51(6):595–607, 2002.

[25] A. Jiang and J. Bruck. *Optimal content placement for en-route web caching*. In Proceedings of the IEEE International Symposium on Network Computing and Applications, pages 9–16, 2003.

[26] K. Poularakis and L. Tassiulas. *Optimal cooperative content placement algorithms in hierarchical cache topologies*. In Proceedings of the Annual Conference on Information Sciences and Systems (CISS), pages 1–6, 2012.

[27] J. Ni and D. H. K. Tsang. *Large-scale cooperative caching and application-level multicast in multimedia content delivery networks*. IEEE Communications Magazine, 43(5):98–105, 2005.

[28] S. Borst, V. Gupta, and A. Walid. *Distributed caching algorithms for content distribution networks*. In Proceedings of the IEEE International Conference on Computer Communications (INFOCOM), pages 1–9, 2010.

[29] J. Zhang. *A literature survey of cooperative caching in content distribution networks*. Computing Research Repository, 1210(71):1–5, 2012.

[30] Z. Li and G. Simon. *In a Telco-CDN, pushing content makes sense*. IEEE Transactions on Network and Service Management (TNSM), 10(3):300–311, 2013.

[31] G. Zhang, Y. Li, and T. Lin. *Caching in information centric networking: A survey*. Computer Networks, 57(16):3128–3141, 2013.

[32] Z. Li and G. Simon. *Time-shifted TV in content centric networks: The case for cooperative in-network caching*. In Proceedings of the IEEE International Conference on Communications (ICC), pages 1550–3607, 2011.

[33] Z. Ming, M. Xu, and D. Wang. *Age-based cooperative caching in information-centric networks*. In Proceedings of the IEEE International Conference on Computer Communications (INFOCOM) Workshops, pages 268–273, 2012.

[34] V. Sourlas, L. Gkatzikis, P. Flegkas, and L. Tassiulas. *Distributed cache management in information-centric networks*. IEEE Transactions on Network and Service Management (TNSM), 10(3):286–299, 2013.

[35] Y. Li, H. Xie, Y. Wen, C.-Y. Chow, and Z.-L. Zhang. *How much to coordinate? Optimizing in-network caching in content-centric networks*. IEEE Transactions on Network and Service Management (TNSM), 12(3):420–434, 2015.

[36] D. Rossi and G. Rossini. *Caching performance of content centric networks under multi-path routing (and more)*. Technical report, Telecom ParisTech, 2011.

[37] D. Tuncer, M. Charalambides, R. Landa, and G. Pavlou. *More control over network resources: an ISP caching perspective*. In Proceedings of the International Conference on Network and Service Management (CNSM), pages 26–33, 2013.

[38] I. Ullah, G. Doyen, G. Bonnet, and D. Gaïti. *A survey and synthesis of user behavior measurements in P2P streaming systems*. IEEE Communications Surveys and Tutorials, 14(3):734–749, 2012.

5

Deadline-aware TCP Congestion Control for Video Streaming Services

M. Claeys, N. Bouten, D. De Vleeschauwer, K. De Schepper, W. Van Leekwijck, S. Latré and F. De Turck.

In proceedings of the IEEE International Conference on Network and Service Management, November 2016.

⋆ ⋆ ⋆

This chapter focuses on the fact that Transmission Control Protocol (TCP), transporting the vast majority of the video traffic, is far from minimizing the number of deadline-missing streams. However, for each byte in a video streaming session, strict delivery deadlines are known at the application layer. By introducing deadline-awareness at the transport layer, video delivery can be optimized by prioritizing specific flows. This chapter proposes a deadline-aware congestion control mechanism, based on a parametrization of the traditional TCP New Reno strategy. By taking into account the available deadline information, the modulation of the congestion window is dynamically adapted to steer the aggressiveness of a considered stream. The proposed approach has been thoroughly evaluated in both a Video-on-Demand (VoD)-only scenario and a scenario where VoD streams co-exist with live streaming sessions and non-deadline-aware traffic. It was shown that in a video streaming scenario the minimal bottleneck bandwidth can be reduced by 16% on average when using deadline-aware congestion control. In co-existence with other TCP traffic, a bottleneck reduction of 11% could be achieved.

5.1 Introduction

Over the past decades, multimedia services have gained a lot of popularity. This growth is largely due to video streaming services, accounting for about 70% of all consumer Internet traffic in 2016 [1]. For delivery of video streaming services, HTTP Adaptive Streaming (HAS) has become the de facto standard. These HTTP-based techniques come with some important advantages. Not only is the video content delivered reliably over Transmission Control Protocol (TCP), HAS also allows seamless interaction through firewalls. On the downside however, as the delivery is based on best-effort Internet, HTTP-based techniques are prone to network congestion and large bandwidth fluctuations due to cross traffic. These influences can be detrimental for the Quality of Experience (QoE).

TCP streaming sessions use a congestion control strategy to avoid congestive collapse. For this purpose, the total number of unacknowledged packets that may be in transit is limited by means of a congestion window at the sender side. It was shown that using this congestion control strategy, multiple flows with similar Round-Trip Times (RTTs) eventually converge to using equal amounts of a contended link [2]. However, different types of services can have different requirements. Considering a video streaming service, hard deadlines are associated with each packet in the video stream once the playout is started. When these deadlines are violated, the playout is temporarily interrupted, negatively impacting the QoE. However, fair bandwidth sharing as introduced by TCP is known to be far from minimizing the number of deadline-missing streams [3].

In general, the client applications are aware of the deadlines associated with the requested content. By introducing these deadlines in the network, multimedia delivery can be optimized by prioritizing specific flows. While DiffServ [4] techniques have been proposed for this purpose in the past, this chapter focuses on a best effort networking scenario by proposing a deadline-aware congestion control strategy for video streaming services, based on the traditional TCP New Reno congestion control mechanism. The proposed strategy only requires changes at the sender side and reduces to TCP New Reno congestion control in absence of deadline information. Furthermore, for deadline-missing streams, the proposed approach falls back to the TCP New Reno congestion control mechanism to avoid congestive collapse. In the proposed approach, deadline information is passed to the transport layer in the request to send data, as is commonly assumed in related work [5–7]. This information provides details about (i) the deadline of the current packet, (ii) the final deadline of the stream and (iii) the bit rate of the streamed content. Based on the deadline information and the current throughput measurement, the algorithm dynamically changes the behavior of the congestion window adaptations. In this way, urgent flows can occasionally achieve a higher throughput than their fair share, while other, less urgent flows back off to allow other sessions to

increase their throughput.

The contributions of this chapter are threefold. First, a parametrization of the congestion avoidance phase of the TCP New Reno congestion control mechanism is proposed. Furthermore, the feasibility of prioritizing streams by changing the configuration of these parameters is demonstrated. Second, a deadline-aware congestion control strategy is introduced, dynamically adapting the parameter configuration of the congestion avoidance phase based on the deadlines of streaming sessions. Third, the performance of the proposed approach is thoroughly evaluated, both using conceptual examples and large scale packet-based simulations in NS-3. For this purpose, both general traffic configurations and Video-on-Demand (VoD)-only scenarios are considered.

The remainder of this chapter is organized as follows. First, Section 5.2 gives an overview of related work on transport protocol optimizations for multimedia delivery. Next, the feasibility of the proposed approach is demonstrated in Section 5.3. The proposed algorithm is introduced in Section 5.4, while its performance is evaluated in Section 5.5. Finally, Section 5.6 presents some final conclusions.

5.2 Related work

Multiple transport protocols that offer novel delivery models to improve the support for multimedia applications have been presented in literature, including Stream Control Transmission Protocol (SCTP) [8], Datagram Congestion Control Protocol (DCCP) [9] and Shared Content Addressing Protocol (SCAP) [10]. However, ossification of the transport layer limits the deployability of new transport protocols. Furthermore, these approaches require changes both at the sending and receiving side, as well as in the network.

TCP-RTM proposes extensions to TCP that improve performance of multimedia applications by allowing a minimal amount of packet re-ordering and loss in the TCP stack [11]. This approach modifies the interaction between the application and the receiver buffer, rather than proposing modifications to TCP itself. Selective negative acknowledgments are used to allow senders to be informed of segments that were skipped by the application, preventing retransmission. Similarly, TCP Hollywood is a protocol offering an unordered, partially reliable message-oriented transport service that is well suited for multimedia applications [12]. While this approach is focused on deployability, the inconsistent retransmission mechanism is visible to middleboxes performing deep packet inspection, which might disrupt these connections. Furthermore, TCP Hollywood requires changes at both the sender and receiver side. While the above approaches focus on flow-based optimization, media-TCP-friendly congestion control (MTCC) takes into account the deadlines for each packet [13]. Furthermore, additional complexity is added to this

solution by introducing a packet-based multimedia model, considering distortion impacts as well as inter-dependencies between multiple packets, using directed acyclic graphs.

In the area of data center services, multiple transport protocols have been proposed with the main objective of minimizing deadline misses. D^3 introduced the idea of including deadline awareness into data center networks by proactively allocating bandwidth before data transmission [5]. PDQ further improves flow completion times compared to D^3 [14]. However, both protocols are incompatible with TCP. Deadline-aware data center TCP (D^2TCP) is a TCP-friendly protocol implementing deadline-aware delivery [6]. Similar to our work, deadline-aware congestion avoidance is implemented by changing the adaptation of the congestion window. In D^2TCP, the window size is modulated based on the deadlines and the extent of congestion in the network. However, as the approach heavily relies on Explicit Congestion Notification (ECN) feedback and is specifically aimed at data center topologies and services, the applicability for video streaming over public Internet is limited. DSTCP builds on the ideas introduced in D^2TCP, adjusting the congestion window size of a flow based on its deadline, its size and the degree of network congestion [7]. As was the case with D^2TCP, DSTCP heavily relies on ECN feedback. L^2DCT has been presented as a TCP-friendly protocol, reducing completion times of short flows in data center networks [15]. As opposed to these works, the focus in this chapter is on multimedia delivery in public Internet, rather than data center networks.

5.3 Feasibility study

As a starting point for the proposed approach, TCP New Reno [16], an improved version of the traditional TCP Reno congestion control mechanism, is used. For a detailed description of the TCP New Reno congestion control mechanism, the reader is referred to literature. In this work, we specifically focus on the congestion avoidance phase of TCP New Reno while the slow start phase, the fast retransmit behavior and the reaction to timeouts are unchanged. This section proposes a parametrization of the congestion avoidance phase and demonstrates the feasibility of steering the aggressiveness of a TCP stream using this parametrized scheme.

5.3.1 Parametrized congestion avoidance

In the congestion avoidance phase, TCP (New) Reno follows an Additive Increase/Multiplicative Decrease (AIMD) scheme to adapt the congestion window size. With AIMD, a linear growth of the congestion window is combined with an multiplicative reduction when a congestion event takes place, resulting in the well known TCP sawtooth behavior. More concretely, with TCP (New) Reno, the con-

gestion window $cwnd$ is increased as shown in equation (5.1) for every incoming non-duplicate acknowledgment, where MSS represents the maximum segment size [17]. This adjustment provides an acceptable approximation to the underlying principle of increasing the congestion window with one full-sized segment every RTT. When congestion is detected in the form of triple duplicate acknowledgments, the congestion window is halved (i.e., $cwnd = 0.5 * cwnd$).

$$cwnd = cwnd + \frac{MSS * MSS}{cwnd} \tag{5.1}$$

It was shown that multiple flows with similar RTTs using the same AIMD congestion control, eventually converge to using equal amounts of a contended link [2]. However, this only holds when these flows use the same AIMD scheme. In this work, we propose to parametrize the AIMD scheme, resulting in an increase of the congestion window as expressed in equation (5.2) for every incoming non-duplicate acknowledgment. Using this equation approximates the increase of the congestion window with one full-sized segment every $\alpha \in \mathbb{R}^+$ RTTs. Upon reception of the third duplicate acknowledgment, the congestion window size is reduced as shown in equation (5.3), for $\beta \in]0; 1[$. For $\alpha = 1.0$ and $\beta = 0.5$, this AIMD scheme corresponds to the scheme used in TCP (New) Reno.

$$cwnd = cwnd + \frac{MSS * MSS}{\alpha * cwnd} \tag{5.2}$$

$$cwnd = (1 - \beta) * cwnd \tag{5.3}$$

5.3.2 Parameter influence

Intuitively, lower values for α or β, causing a faster increase or slower decrease of the congestion window size respectively, result in more aggressive behavior and corresponding higher throughput. However, to study the influence of these parameters in detail, simulations have been performed to compare the throughput of a client using TCP New Reno congestion control to a client using different parameter values.

For this purpose, two scenarios have been considered where respectively two and ten clients simultaneously try to send data at a rate of 10Mbps over a bottleneck link with a capacity equal to half of the sum of requested rates (i.e. 10Mbps and 50Mbps in the two and ten client scenario respectively). To estimate the influence of the value of α, for one of the clients α is varied between 0.05 and 5.0 with a fixed value of $\beta = 0.5$, while the other client(s) use TCP New Reno congestion control (i.e. $\alpha = 1.0, \beta = 0.5$). Similarly, the influence of β is evaluated by varying its value between 0.01 and 0.99 while keeping a fixed value of α=1.0. Due to the probabilistic nature of the applied Random Early Detection (RED) queuing

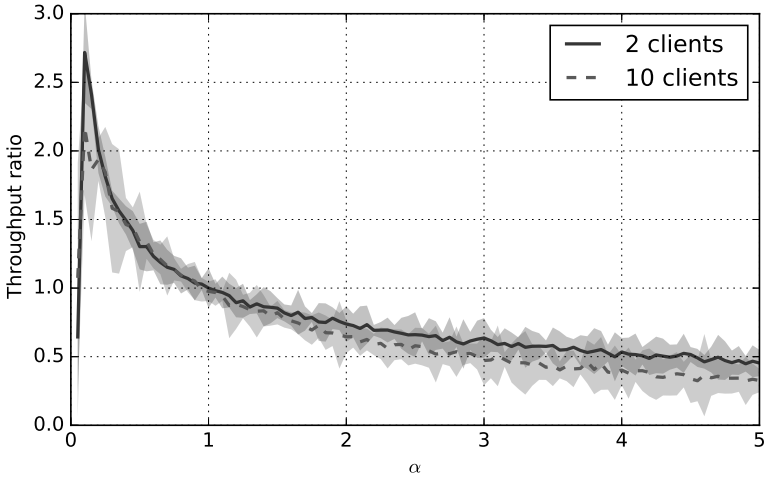


Figure 5.1: Influence of the AIMD parameter α on the average achieved relative throughput on the contended link. The areas represent the 95% confidence interval.

discipline [18], all experiments have been performed for five iterations, presenting the average results.

Figure 5.1 shows the influence of the value of α on the ratio between of the throughput achieved by the client using varied parameters and the average throughput achieved by the other clients, using TCP New Reno congestion control. As expected, both in the two and ten client scenario, for low values of α (< 1.0) the achieved throughput is significantly higher than the fair share throughput while the opposite is true for high values of α (> 1.0). It can be seen that for very small values of α (< 0.10), the client becomes too aggressive, resulting in decreased performance. By increasing the congestion window too fast, the client perceives a lot of timeouts, resulting the stream to execute in the slow start phase very often. Furthermore, the throughput ratio converges and the influence is limited when increasing α above 4.0.

Furthermore, Figure 5.2 shows the influence of the value of α on the total throughput on the contended link when all clients use this specific value, relative to the throughput achieved with TCP New Reno congestion control. It can again be seen that for very small values of α (< 0.1), the clients become too aggressive, causing a fallback to the slow start phase too often. As a result, the flows are not able to fill the link. For values above $\alpha >= 0.5$, the general performance is within 1% of the TCP New Reno performance.

Similarly, the influence of the value of β is presented in Figure 5.3 and Figure 5.4. It can again be seen that, as expected, a flow is able to achieve a significantly higher or lower throughput than its fair share by respectively using a value

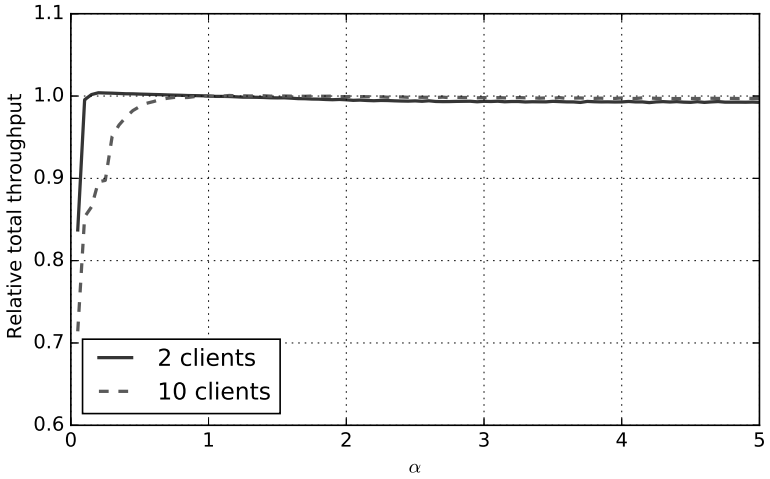


Figure 5.2: Influence of the AIMD parameter α on the total throughput achieved on the contended link, relative to the throughput achieved with TCP New Reno congestion control.

of $\beta < 0.5$ or $\beta > 0.5$. As was the case with α, a general performance degradation can be perceived when all streams become too aggressive (i.e. $\beta < 0.1$). Furthermore, a performance drop of about 10% can be noticed when all streams use high values of β. As this situation only occurs when all streams voluntarily back off, indicating that the available bandwidth is higher than required for all flows, it is unlikely to occur in practice.

Based on the above analysis, it can be summarized that the aggressiveness of a stream can effectively be influenced by varying the value of α between 0.1 and 4.0 or varying the value of β between 0.1 and 0.95.

5.4 Algorithm

As shown in Section 5.3, the throughput can be increased or decreased by using different values for α or β. In this section, an algorithm is proposed to dynamically adapt the parameter values based on the deadlines of a streaming session. For this purpose, the concept of *deadline margin* is introduced. Consider a video stream s with a bit rate r_s and a total length of l_s seconds. When this stream has a begin deadline d_0, for every byte $b \in [0; \frac{r_s * l_s}{8}[$ the corresponding deadline d_b can be calculated as shown in equation (5.4). The deadline of the last byte of the stream is called the end deadline of the stream, denoted as $d_e = d_0 + l_s$.

$$d_b = d_0 + \frac{b * 8}{r_s} \tag{5.4}$$

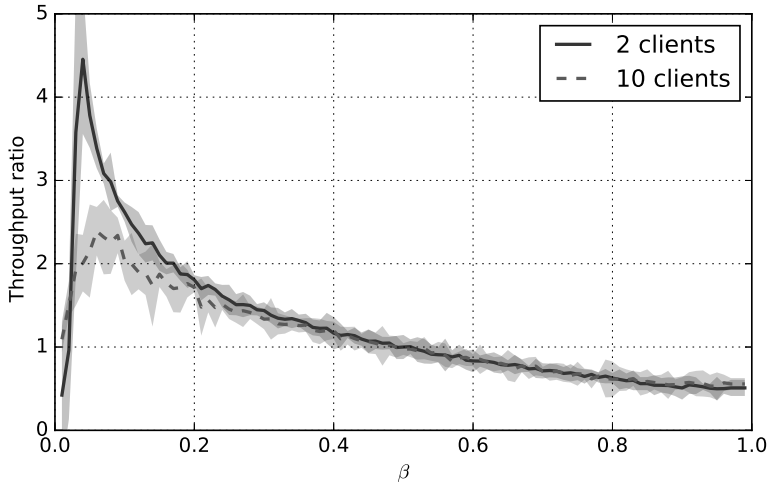


Figure 5.3: Influence of the AIMD parameter β on the average achieved relative throughput on the contended link. The areas represent the 95% confidence interval.

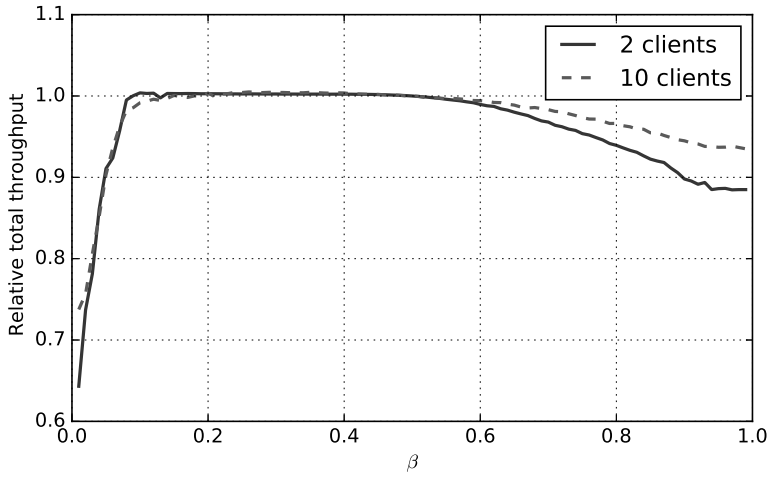


Figure 5.4: Influence of the AIMD parameter β on the total throughput achieved on the contended link, relative to the throughput achieved with TCP New Reno congestion control.

When at time t the next byte to be sent is b, the current deadline margin is denoted as $m = d_b - t$. The general goal of the proposed approach is to keep this deadline margin between a lower and upper bound, denoted as m_l and m_u respectively. Both m_l and m_u are parameters of the algorithm which are configured a priori. In order to keep the deadline margin within the predefined bounds, the value of α or β can be changed within their respective ranges $[\alpha_l; \alpha_u]$ and $[\beta_l; \beta_u]$, with a granularity of α_g and β_g respectively. As discussed in Section 5.3, it is suggested to vary α between 0.1 and 4.0 at a granularity of 0.1, or to vary β between 0.1 and 0.95 at a granularity of 0.05. To avoid too much fluctuation in the parameter values and to reduce the computational overhead, it is suggested to only perform parameter updates at fixed time intervals, indicated by the reactivity time t_r. Based on preliminary evaluations, a reactivity time of t_r=1s will be used in the remainder of this chapter.

Algorithm 5.1 presents the pseudo-code of the proposed algorithm to dynamically adapt the parameter value α. In an identical way, the algorithm can be applied to dynamically change the value of β. This algorithm is executed every t_r seconds, as long as no deadlines were missed for the considered stream. To avoid congestive collapse and to maintain a degree of fairness with other TCP flows, a streaming session permanently falls back to TCP New Reno congestion control (i.e. α=1.0, β=0.5) when a deadline was missed. Based on the current time t and the deadline d_b for the next byte to send b, the current margin m is calculated (line 1). When the current margin is out of the predefined bounds, the value of α is directly updated to the highest (α_u) or lowest (α_l) value to drastically decrease or increase the aggressiveness of the stream respectively (lines 2-5). When the current margin is between the bounds, the upper and lower threshold of the required throughput, T_u and T_l, are calculated (lines 7-10). T_u and T_l respectively denote the highest and lowest required throughput that allows to finish the remainder of the stream without leaving the deadline margin bounds. Their calculation is based on the remaining time rt until the final deadline and the remaining number rb of bytes to send. When the current throughput measurement T is above T_u or below T_l, the deadline margin is expected to grow above m_u or shrink to less than m_l before the end of the stream is reached. To avoid this, the difference m_δ between the current deadline margin and the approaching bound is calculated, as well as the remaining part α_Δ of the range of α values that allows us to change the aggressiveness in the required direction (lines 11-20). Based on these values, the critical time period C by which leaving the margin bounds is expected, can be calculated (line 21). In combination with the remaining range α_Δ and the time between consecutive parameter changes t_r, this critical time period is used to calculate the degree to which the value of α should be changed, rounded to the change granularity α_g (lines 22-23).

To clarify the rationale behind the algorithm, an illustrative example is pre-

Input:

t: current time
T: current throughput measurement
d_b: deadline of next byte to send
d_e: end deadline of the stream
r_s: bit rate of the stream

1: $m = d_b - t$
2: **if** $(m < m_l)$ **then**
3: $\quad \alpha = \alpha_l$
4: **else if** $(m > m_u)$ **then**
5: $\quad \alpha = \alpha_u$
6: **else**
7: $\quad rt = d_e - t$
8: $\quad rb = (d_e - d_b) * r_s$
9: $\quad T_u = \frac{rb}{rt - m_u}$
10: $\quad T_l = \frac{rb}{rt - m_l}$
11: $\quad$ **if** $(T > T_u)$ **then**
12: $\quad\quad m_\delta = m_u - m$
13: $\quad\quad \alpha_\Delta = \alpha_u - \alpha$
14: $\quad$ **else if** $(T < T_l)$ **then**
15: $\quad\quad m_\delta = m - m_l$
16: $\quad\quad \alpha_\Delta = \alpha_l - \alpha$
17: $\quad$ **else**
18: $\quad\quad m_\delta = \infty$
19: $\quad\quad \alpha_\Delta = 0$
20: $\quad$ **end if**
21: $\quad C = \frac{m_\delta * r_s}{|r_s - T|}$
22: $\quad \alpha_\delta = \text{round}(\frac{\alpha_\Delta * t_r}{C}; \alpha_g)$
23: $\quad \alpha = \alpha + \alpha_\delta$
24: **end if**

Algorithm 5.1: Outline of the proposed algorithm to dynamically adapt the parameter value α based on the current deadline margin. This update is performed every t_r seconds.

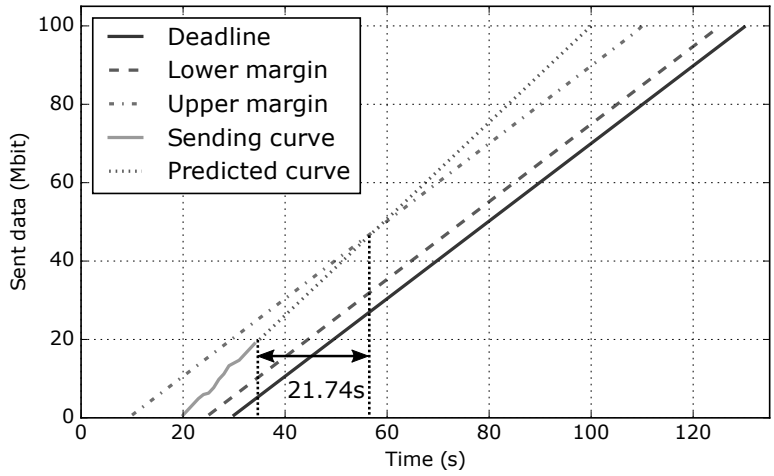
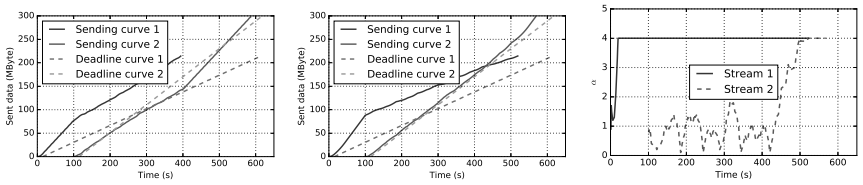


Figure 5.5: Graphical illustration of the rationale behind the proposed approach.

sented in Figure 5.5. In this example, we consider a video streaming session s with begin deadline d_0=30s, a total duration l_s=100s and a bit rate b_s=1Mbps. The presented sending curve and deadline curve respectively show the time when each byte is sent and when it is required. The streaming session started at 20s, the margin bounds are configured to m_u=20s and m_l=5s and the reactivity interval t_r is set to 1s. At time t=35s, 20Mbit has been sent already and the deadline margin amounts to about m=15s. At this point in time, rb=80Mbit remains to be sent in the next rt=95s. To be able to finish the stream without exceeding the margin bounds, the required throughput should be between $T_l = \frac{80\text{Mbit}}{90\text{s}} = 0.89\text{Mbps}$ and $T_u = \frac{80\text{Mbit}}{75\text{s}} = 1.07\text{Mbps}$. Given the current throughput measurement T=1.23Mbps and the current distance from the upper bound m_δ=5s, the deadline margin is expected to exceed the margin upper bound in $C = \frac{5\text{Mbit}}{0.23\text{Mbps}} = 21.74\text{s}$. Given the reactivity interval t_r=1s, the current estimate results in 21 remaining chances to adapt α to reduce the aggressiveness. When currently $\alpha = 1.0$, α will be increased by $\alpha_\delta = \text{round}(\frac{(4.0-1.0)*1.0}{21.74}; 0.1) = 0.1$, resulting in α to be set to 1.1. When the difference between the current throughput and the required throughput would be higher, the change in α value would be more significant as the time to react would be shorter.

5.5 Evaluation

To evaluate the performance of the proposed approach, we first demonstrate the effectiveness of dynamically changing the AIMD parameters in a scenario with two sessions in Section 5.5.1. Next, Section 5.5.2 will evaluate the proposed ap-



(a) Performance of two VoD streaming clients using TCP New Reno congestion control.

(b) Performance of two VoD streaming clients using deadline-aware congestion control.

(c) Adaptive behavior of the value of α over time based on the current deadline information.

Figure 5.6: Conceptual demonstration of the proposed deadline-aware congestion control mechanism in a scenario with two VoD streaming sessions.

proach in larger scenarios using real-world characteristics. All simulations have been performed in NS-3 using a setup where all considered streams share a common bottleneck link on which RED queue management is applied.

5.5.1 Conceptual demonstration

To show the benefits of using deadline-aware congestion control, a small-scale scenario is considered where two VoD streaming sessions share a bottleneck link with a capacity of 10Mbps and a delay of 30ms. The first stream has a bit rate of 3Mbps, has a begin deadline of 15s after the initial request and a total duration of 600s. After 100s, the second stream with a bit rate of 5Mbps is initiated with a begin deadline of 115s relative to the experiment start and a total duration of 500s.

Figure 5.6a shows the sending curves and the corresponding deadline curves of both video streams in a scenario where TCP New Reno congestion control is applied (α=1.0, β=0.5). It can be seen that during the first 100s, the first stream can use the full capacity of the link and can deliver all bytes long before their deadline. After 100s, the second stream starts and both streams get an equal share of the link capacity. However, this bandwidth share is not sufficient for the second stream to deliver its data before the deadline, resulting in deadline misses for over 70% of the data. In the worst case, the data is delivered only 50s after its deadline. However, the throughput for the first stream exceeds the required bit rate, causing the entire stream to be delivered about 200s earlier than the final deadline.

Figure 5.6b shows the performance of the proposed deadline-aware congestion control where the α parameter is dynamically changed for the same scenario, with the deadline margin bounds set to m_l=5s and m_u=20s. The resulting behavior of α is shown in Figure 5.6c. It can again be seen that during the initial 100s, the first stream can use the full capacity of the link and can deliver all bytes long before their respective deadlines. As the achieved throughput of that stream is signifi-

cantly higher than the bit rate of the video, the α value quickly increases to its maximum value of 4.0 to become less aggressive and free up bandwidth for other streams if required. When the second stream starts, its fair share bandwidth part does not suffice to deliver the stream in time. Therefore, the second stream becomes more aggressive by lowering its α value. To keep the throughput within the predefined margin without using more bandwidth than required, α starts fluctuating between 0.1 and 1.5. When the first stream finishes, the second stream can use the entire capacity, resulting in a higher throughput than required. Therefore, the second stream backs off by increasing its α value. In this scenario, using deadline aware congestion control allows both streams to deliver all bytes in time.

5.5.2 Larger scale evaluations

To evaluate the impact of the proposed approach in a scenario with multiple (deadline-aware) streams, several scenarios have been generated using realistic network traffic compositions. In each of these scenarios, the minimum bottleneck bandwidth, required to finish all streams without deadline misses, is established for both TCP New Reno congestion control and the deadline-aware congestion control. For this purpose, a binary search strategy has been applied to find the bottleneck bandwidth with a granularity of 100kbps. Given the probabilistic nature of the RED queue management, all simulations have been performed for 3 iterations. First, Section 5.5.2.1 considers a setup with only deadline-aware VoD streams, while more general setups, including deadline-aware live and VoD streaming sessions as well as non-deadline-aware traffic, are considered in Section 5.5.2.2.

5.5.2.1 VoD-only scenarios

To evaluate the performance of the proposed approach, VoD request traces have been constructed based on statistics provided by Conviva. In October 2015, Conviva opened access to viewer experience data, based on the analysis of 4 billion video streams per month spread across 180 countries[1]. According to this dataset, in the first quarter of 2016, the average bit rate of a VoD stream amounts to 2.4Mbps. Taking into account general adaptive streaming characteristics, each streaming session in the constructed request traces was assigned a uniformly distributed bit rate between 0.9Mbps and 3.9Mbps. The duration (in minutes) of each session can be modeled using a log-normal distribution with μ=2.2 and σ=1.5 [19, 20]. While the number of sessions is varied between 25 and 100, in each scenario the start times of the video streams are uniformly distributed over a period of 60minutes. In this way, different levels of load are considered. For each session, the sending of data can be started at most m_0 seconds before the begin deadline d_0 (i.e. at time $t = d_0 - m_0$). All flows share a bottleneck link with a delay of 30ms.

[1]Conviva dataset - http://www.conviva.com/industry-data-portal/

Table 5.1: Evaluated parameter configurations.

Parameter	Values
Number of sessions	25, 50, 75, 100
m_l	0, 5, 10
m_u	10, 20
m_0	0.5, 5, 10, 20

Multiple parameters have been evaluated, as presented in Table 5.1. For each configuration, the approach has been evaluated in 10 scenarios, randomly generated according to the above characteristics. To assess the performance of the proposed approach, in each of the scenarios the minimal bottleneck bandwidth B_{da} that is required to deliver all streaming sessions without deadline misses using deadline-aware congestion control is defined by simulation and compared to the same bottleneck bandwidth B_{nr} required when using TCP New Reno congestion control. Furthermore, the theoretical lower bound for the bottleneck bandwidth B_{edf} when using the Earliest Deadline First (EDF) policy, scheduling the data with the earliest deadline amongst all flows at any point in time, is calculated. It is important to note that this optimal solution cannot be achieved in practice and does not take into account TCP overhead. Therefore, the actual lower bound will be significantly higher than B_{edf}, which serves as a benchmark. Based on these values, the metric μ is defined as shown in equation (5.5), representing the ratio between the achieved gain and the optimal achievable gain.

$$\mu = \frac{B_{nr} - B_{da}}{B_{nr} - B_{edf}} \tag{5.5}$$

The influence of the number of streaming sessions on the performance of the proposed deadline-aware congestion control mechanism, dynamically adapting the value of α, is presented in Figure 5.7 for different configurations of m_l and m_u. It can be seen that for each configuration of the deadline margin bounds, the performance of the proposed approach increases with the number of streaming sessions. As the contention is limited with fewer streams, the benefits of using deadline-aware congestion control are narrow. However, when the number of streaming sessions increases, the performance of the proposed approach significantly increases as well, saturating around μ=0.60. While the performance difference between the configurations is limited, m_l=5s and m_u=20s yields the best performance on average. When margin bounds are set closer to the deadline, the reaction time for the algorithm increases, resulting in a slightly lower performance.

Similarly, Figure 5.8 shows the performance of the proposed deadline-aware congestion control mechanism dynamically adapting the value of β. It can be seen that for each configuration, the performance is lower compared to when dynamically adapting α. The explanation behind this finding is in the role of both param-

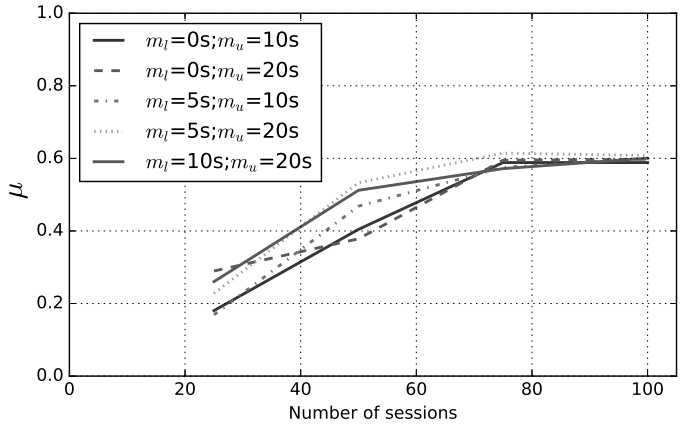


Figure 5.7: Performance of the deadline-aware congestion control mechanism by dynamically adapting the value of α for multiple parameter configurations in a VoD-only scenario.

eters in the congestion control mechanism. While the value of α has an impact for every received acknowledgment, the value of β only impacts the behavior upon receiving triple duplicate acknowledgments in the event of congestion. Therefore, the impact of dynamically adapting β on the congestion window is perceived less frequently. Furthermore, the effect of changing the value of β cannot be perceived immediately, but only at the next congestion event. Based on this analysis, in the remainder of this chapter the focus will be on the adaptation of the α parameter.

The results presented in Figure 5.7 show that for higher network loads, the proposed approach can yield around 60% of the theoretical upper bound for the achievable gain (i.e. μ=0.60) by dynamically adapting the value of α. To compare the performance of the deadline-aware congestion control with the TCP New Reno congestion control, Figure 5.9 presents the ratio between the minimal bottleneck bandwidth required to streams all sessions without deadline misses, i.e. $\frac{B_{da}}{B_{nr}}$, and the corresponding standard deviation. Based on the above evaluation, the deadline margin bounds were set to m_l=5s and m_u=20s, considering multiple initial deadline margins m_0. It can be seen that for a low number of sessions, the minimal bottleneck bandwidth required to stream all sessions without deadline misses is 8% lower compared to using TCP New Reno congestion control on average, while for short initial deadline margins (m_0=0.5s) no gain can be achieved. For a higher number of sessions, the bottleneck bandwidth reduction amounts to between 15% and 23%. Furthermore, it can be seen that the influence of the initial deadline margin m_0 on the average performance is insignificant for a higher number of sessions. However, higher initial margins result in a more consistent performance increase, as presented by the decreasing standard deviations. In general, an average bottle-

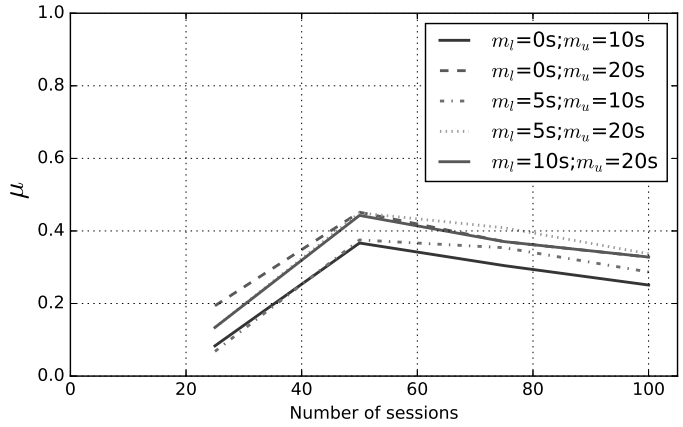


Figure 5.8: Performance of the deadline-aware congestion control mechanism by dynamically adapting the value of β for multiple parameter configurations in a VoD-only scenario.

neck bandwidth reduction of 16% is achieved.

5.5.2.2 General scenarios

In a general scenario, besides VoD streaming sessions, other types of traffic are present in the network as well. According to Cisco, in 2016 70% of all consumer Internet traffic consists of video streaming [1]. Out of the Conviva dataset it can be deduced that 36% of all streamed video data consists of live video, while the remaining 64% can be considered as VoD. Combining this information results in a traffic pattern where 44% of the network traffic consists of VoD, 26% consists of live streaming and the remaining 30% is considered as non deadline sensitive traffic. Given the size difference between the different types of streams, it is important to note that this division accounts to the amount of data streamed for each type, rather than the number of sessions for each type.

For the VoD sessions, the same characteristics as in the previous section have been used. To generate the live streaming sessions, a uniformly distributed bit rate between 1.1Mbps and 4.1Mbps has been used, based on the average bit rate of 2.6Mbps as reported by Conviva. Based on the literature, the duration (in seconds) of these sessions is modeled using a log-normal distribution with μ=5.19 and σ=1.44 [20, 21]. As the playout of the live stream is considered to be 10s behind of the live signal, data can only be send at most 10s before its deadline. Therefore, the deadline margin m can never exceed 10s for live streaming sessions. For the non deadline sensitive traffic, regular file transfers with a uniformly distributed size between 0.5 and 600Mbyte have been generated.

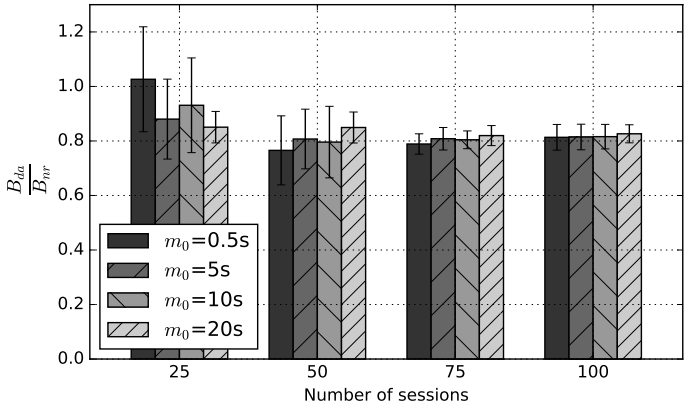


Figure 5.9: Relative performance of the deadline-aware congestion control by dynamically adapting α for m_l=5s, m_u=20s and multiple initial deadline margins m_0 in a VoD-only scenario.

Given the absence of deadline information for file transfers, the theoretical optimal gain when using EDF can not be defined. All non-deadline aware traffic has an infinite deadline, causing them to only be scheduled by EDF when no other deadline-aware traffic is present. As this does not allow a fair comparison, for the general scenarios the performance of the deadline-aware congestion control will be compared only to the TCP New Reno congestion control mechanism.

Figure 5.10 shows the relative performance of the deadline-aware congestion control compared to the TCP New Reno congestion control. Based on the results presented in Section 5.5.2.1, the value of α was dynamically adapted, using deadline margin bounds set to m_l=5s and m_u=20s. It can be seen that on average, a bandwidth reduction of between 5% and 18% can be achieved. As the file transfers cannot benefit from the deadline-awareness, the relative performance gain is lower compared to a VoD-only scenario. Furthermore, as the live streaming sessions can at most be 10s ahead of their deadlines, the potential benefits of deadline-aware congestion control are limited as well. However, even though only 44% of the traffic can fully take advantage of the deadline-aware congestion control, the minimal bottleneck bandwidth can be reduced by 11% on average compared to using TCP New Reno congestion control.

To analyze the impact of the deadline-aware congestion control strategy on the performance of regular TCP traffic, the notation F_{da} is introduced representing the sum of the throughput achieved by each of the regular file transfers in a scenario where deadline-aware congestion control is used. Similarly, the sum of the throughput of all regular file transfers when using TCP New Reno congestion control in the same scenario is denoted as F_{nr}. Based on these values, the ratio $\frac{F_{da}}{F_{nr}}$

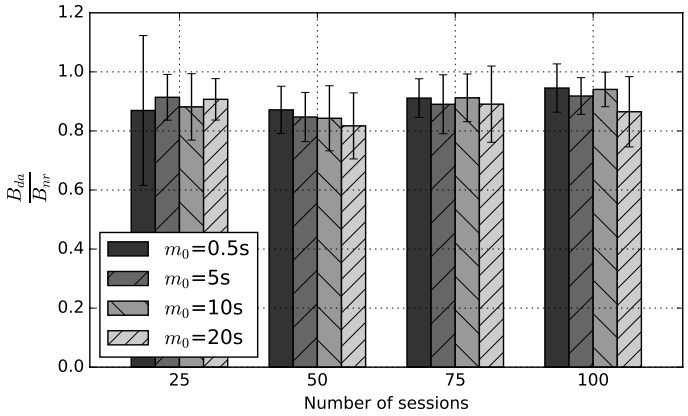
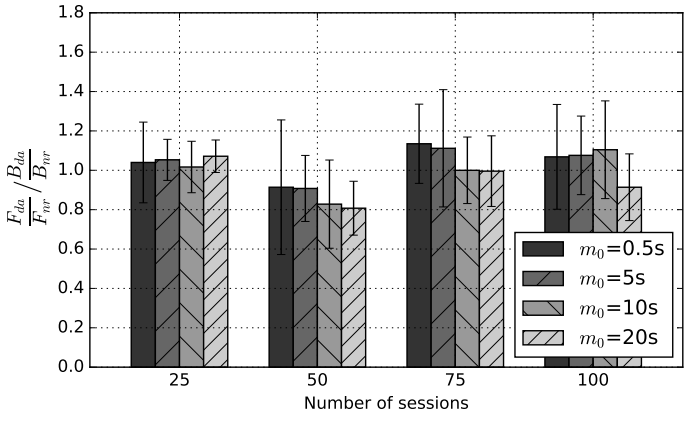


Figure 5.10: Relative performance of the deadline-aware congestion control by dynamically adapting α for m_l=5s, m_u=20s and multiple initial deadline margins m_0 in a general traffic scenario.

shows the relative increase (> 1) or decrease (< 1) in throughput for non-deadline-aware traffic. In Figure 5.11, this value is compared to $\frac{B_{da}}{B_{nr}}$. In this graph, a ratio of 1 indicates that the change in throughput for the non-deadline-aware traffic is of the same relative magnitude as the change in bottleneck bandwidth. As a result, the relative bandwidth share of the non-deadline-aware traffic is unchanged. It can be seen that only in the scenarios with 50 sessions, the relative bandwidth share of the non-deadline-aware traffic is significantly reduced. In general however, with an average ratio of 1.00, the fairness with regular TCP traffic is maintained.

5.6 Conclusions

In this chapter, a deadline-aware congestion control strategy was presented based on the congestion avoidance phase of the TCP New Reno congestion control mechanism. By introducing deadline information at the transport layer, the modulation of the congestion window can be dynamically adapted to minimize the number of deadline-missing flows. The proposed approach only requires changes at the sender side and is fully transparent in the network. The deadline-aware congestion control mechanism has been thoroughly evaluated in both a VoD-only scenario and in co-existence with live streaming sessions and regular non-deadline-sensitive TCP traffic. It was shown that in a VoD scenario, the minimum bottleneck bandwidth required to finish all streaming sessions without deadline misses could be reduced by 16% on average. When considering more general scenarios, an average bottleneck reduction of 11% was achieved while maintaining a fair bandwidth share for regular TCP traffic.



p

Figure 5.11: Impact of the deadline-aware congestion control mechanism by dynamically adapting α for m_l=5s, m_u=20s on the performance of non-deadline-aware traffic in a general traffic scenario.

Acknowledgment

M. Claeys and N. Bouten are funded by a grant of the Agency for Innovation and Entrepreneurship in Flanders (VLAIO). The research was performed partially within the ICON SHIFT-TV project (under grant agreement no. 140684). This work was partly funded by FLAMINGO, a Network of Excellence project (318488) supported by the European Commission under its Seventh Framework Programme.

References

[1] Cisco. *Cisco Visual Networking Index: Forecast and methodology, 2014-2019*. Technical report, Cisco, 2015.

[2] D. M. Chiu and R. Jain. *Analysis of the increase and decrease algorithms for congestion avoidance in computer networks*. Computer Networks and ISDN Systems, 17(1):1–14, 1989.

[3] N. Bansal and M. Harchol-Balter. *Analysis of SRPT scheduling*. In Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS), volume 29, pages 279–290, 2001.

[4] K. Nichols, S. Blake, F. Baker, and D. Black. *Definition of the Differentiated Services field (DS field) in the IPv4 and IPv6 headers*. Network Working Group RFC 2474, pages 1–20, 1998.

[5] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowstron. *Better Never than Late: Meeting Deadlines in Datacenter Networks*. In Proceedings of the ACM International Conference on Communications Architectures, Protocols and Applications (SIGCOMM, pages 50–61, 2011.

[6] B. Vamanan, J. Hasan, and T. Vijaykumar. *Deadline-aware datacenter TCP (D2TCP)*. ACM SIGCOMM Computer Communication Review, 42(4):115–126, 2012.

[7] G. Li, Y. Xu, and D. Cui. *A deadline and size aware TCP scheme for datacenter networks*. In Proceedings of the IEEE International Conference on Communication Technology (ICCT), pages 366–371, 2013.

[8] R. Stewart. *Stream Control Transmission Protocol*. Network Working Group RFC 4960, pages 1–152, 2007.

[9] E. Kohler, M. Handley, S. Floyd, and J. Padhye. *Datagram Congestion Control Protocol (DCCP)*. Network Working Group RFC 4340, pages 1–130, 2006.

[10] K. De Schepper, B. De Vleeschauwer, C. Hawinkel, W. Van Leekwijck, J. Famaey, W. Van de Meerssche, and F. De Turck. *Shared Content Addressing Protocol (SCAP): Optimizing multimedia content distribution at the transport layer*. In Proceedings of the IEEE Network Operations and Management Symposium (NOMS), pages 302–310, 2012.

[11] S. Liang and D. Cheriton. *TCP-RTM: Using TCP for real time applications*. In Proceedings of the IEEE International Conference on Network Protocols (ICNP), pages 1–20, 2002.

[12] S. McQuistin, C. Perkins, and M. Fayed. *TCP goes to Hollywood*. In Proceedings of the ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), pages 1–6, 2016.

[13] H. P. Shiang and M. Van Der Schaar. *A quality-centric TCP-friendly congestion control for multimedia transmission*. IEEE Transactions on Multimedia, 14(3):896–909, 2012.

[14] C.-Y. Hong, M. Caesar, and P. B. Godfrey. *Finishing flows quickly with preemptive scheduling*. In Proceedings of the ACM International Conference on Applications, technologies, architectures, and protocols for computer communications, pages 127–138, 2012.

[15] A. Munir, I. A. Qazi, Z. A. Uzmi, A. Mushtaq, S. N. Ismail, M. S. Iqbal, and B. Khan. *Minimizing flow completion times in data centers*. In Proceedings of the IEEE International Conference on Computer Communications (INFOCOM), pages 2157–2165, 2013.

[16] T. Henderson, S. Floyd, A. Gurtov, and Y. Nishida. *The New Reno modification to TCP's fast recovery algorithm*. Internet Engineering Task Force RFC 6582, pages 1–16, 2012.

[17] M. Allman, V. Paxson, and W. Stevens. *TCP congestion control*. Network Working Group RFC 2581, pages 1–14, 1999.

[18] S. Floyd and V. Jacobson. *Random early detection gateways for congestion avoidance*. IEEE/ACM Transactions on Networking, 1(4):397–413, 1993.

[19] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng. *Understanding user behavior in large-scale video-on-demand systems*. ACM SIGOPS Operating Systems Review, 40(4):333–344, 2006.

[20] I. Ullah, G. Doyen, G. Bonnet, and D. Gaïti. *A survey and synthesis of user behavior measurements in P2P streaming systems*. IEEE Communications Surveys and Tutorials, 14(3):734–749, 2012.

[21] E. Veloso, V. Almeida, W. Meira, A. Bestavros, and S. Jin. *A hierarchical characterization of a live streaming media workload*. IEEE/ACM Transactions on Networking, 14(1):133–146, 2006.

6 Conclusions and Perspectives

In this dissertation, multiple contributions in the area of management of video streaming services have been presented. The proposed management solutions optimize the quality of the delivered service by focusing on different locations in the delivery chain. The optimizations aim at improving the quality as perceived by the end-users, denoted as Quality of Experience (QoE), as well as at reducing the strain imposed on the underlying network resources. This concluding chapter summarizes how this dissertation addressed multiple problems currently observed in video streaming services and identifies interesting challenges to be addressed by the research community.

6.1 Review of problem statements

Each of the solutions proposed in the previous chapters addressed one of the problem statements presented in Chapter 1 as follows:

1. *Existing HTTP Adaptive Streaming (HAS) rate adaptation heuristics are tailored to specific network configurations.* Therefore, these algorithms are unable to cope with a vast range of highly dynamic network settings. In Chapter 2, a novel Reinforcement Learning (RL)-based HAS client is proposed that is able to dynamically adjust its behavior to the perceived networking environment. By introducing the current buffer filling level and the available bandwidth as the state of the Q-learning technique, the client is able to continuously optimize the delivered service quality.

A frequency-adjusted extension of the Q-learning technique was proposed to further increase the performance of the proposed approach. Furthermore, by incorporating HAS domain knowledge in the initial model, the learning phase can be significantly reduced. Since previously obtained knowledge can be reused when learning on a new video sequence, the learning agent can be trained offline in a vast range of scenarios before being applied in an online HAS client. Thorough evaluations have shown that using the self-learning approach, the QoE can be increased by 11% to 18% in terms of estimated Mean Opinion Score (eMOS) in different bandwidth environments, compared to traditional deterministic algorithms. Furthermore, the average video freezing time could be reduced by up to 67%.

2. *Current video delivery over Content Delivery Networks (CDNs) exerts a lot of pressure on Internet Service Provider (ISP) networks.* As this results in increasing operating costs and decreasing revenues, ISPs have started to deploy so-called telco-Content Delivery Networks (telco-CDNs), given them more control over their network resources. In Chapter 3, a scenario was proposed where the telco-CDN infrastructure is virtualized, allowing the storage capacity to be leased to multiple third parties such as content or service providers.

   Based on this scenario, a hybrid multi-tenant cache management system was proposed, combining proactive capacity allocation and content placement with traditional reactive caching strategies. By periodically changing the capacity allocation and content placement based on the predicted content popularity and its geographical distribution, the system is able to reduce the bandwidth usage inside the ISP network while simultaneously reducing the load on the origin server. On top of that, the reactive caching capacity is able to react to unexpected changes in the request pattern.

   Using a request trace of the Video-on-Demand (VoD) service of a leading European telecom operator, it was shown that the number of requests server from within the ISP network could be increased by 19% and 43% compared to purely proactive and reactive approaches, respectively. Simultaneously, the load on the underlying network resources could be reduced by up to 7%. However, this bandwidth reduction also considers the migration overhead during off-peak hours. When considering only the video streaming traffic, a bandwidth reduction of more than 10% can be achieved, significantly reducing the server load and backhaul traffic. Furthermore, the proposed approach was shown to store the content up to 8% closer to the end-user, positively influencing the perceived service quality by reducing the total delay.

3. *The management of video streaming services does not fully take into account the changing user behavior.* Chapter 4 focused on the phenomenon of binge

watching, where users frequently watch multiple consecutive episodes of the same series in one sitting. By taking into account this recent trend in user behavior, with a significant probability it can be assumed that when a user is watching a specific episode of a series, the following episode will be watched thereafter. Based on this assumption, the accuracy of future reuse times estimates for video segments can be significantly increased.

Two cache replacement strategies for segmented video content have been proposed, based on these reuse time estimates. It was shown that by announcing future video streaming sessions based on the binge watching phenomenon, the cache hit ratio could be increased by up to 10%. Additionally, coordination within the caching network has been introduced using an election-based strategy. It was shown that this results in an additional hit ratio increase of up to 12%. Combining these factors results in a hit ratio increase of 22% compared to the state-of-the-art. This hit ratio increase leads to a reduction of 7% in terms of server load and backhaul traffic.

4. *Fair bandwidth sharing introduced by Transmission Control Protocol (TCP) is known to be far from minimizing the number of deadline misses.* However, deadline misses are known to have a detrimental impact on the QoE for video streaming services. In Chapter 5, it was proposed to introduce the deadline information associated with video streams, currently only known at the application layer, into the network. A parametrization of the TCP New Reno congestion control strategy has been presented, based on which a deadline-aware congestion avoidance algorithm was proposed. By dynamically adapting the modulation of the congestion window, the number of deadline-missing flows can be reduced. Evaluations have shown that in a non-adaptive VoD-only scenario, the bottleneck bandwidth required to finish all video streams without deadline misses can be reduced by 16% on average. In a scenario where non-video traffic is considered as well, a bottleneck bandwidth reduction of 11% can be achieved, without negatively influencing other traffic. This bandwidth reduction can significantly decrease the investment frequency for a network operator.

   Appendix A elaborated on this work by introducing a congestion-aware extension of the deadline-aware algorithm. By taking into account the current level of congestion, the performance of the algorithm can be further improved. Furthermore, the performance gain in terms of QoE has been evaluated for both of the proposed algorithms. It was shown that, depending on the level of congestion, an the eMOS could be increased by up to 11% on average, mainly caused by a reduction of 94% in terms of video freezes.

6.2 Future perspectives

This dissertation proposed multiple solutions to deal with different problems that are currently perceived when delivering video streaming services. However, given the vast popularity of video streaming services, the strong trend of continuous innovations over the last decade is expected to continue in the years to come, both in terms of service offerings and the underlying technology. This will result in new challenges to be faced by the research community. In this section, four important future research perspectives are identified and discussed.

6.2.1 Mobile video streaming

Over the last years, mobile devices such as smartphones and tablet have been increasingly used to access the Internet. In 2015, the global mobile data traffic has grown with 74%. Importantly, 55% of that traffic was accounted to mobile video streaming. While most of the approaches proposed in this dissertation can be applied to mobile video streaming as well, some additional challenges are introduced. For example, the strong mobility causes the mobile devices to commonly shift between access points and access technologies (e.g., 3G, 4G, WiFi) while the user is moving. To avoid impacting the QoE, these handovers should be seamless and not interrupting the video playout. Furthermore, research in this area would greatly benefit from the availability of realistic network characteristics models or measurement studies in terms of mobile connectivity. Within the research group, some first initiatives in this area have already been reported [1].

Additionally, these mobile devices typically have a limited battery capacity. Furthermore, multiple variables such as the used access technology and the played video quality can significantly influence the battery usage of the video streaming session. Therefore, novel QoE estimation models are required to evaluate the influence of battery depletion on the user experience, as well as rate adaptation algorithms taking into account this information.

6.2.2 QoE fairness between HAS clients

As in HAS the video segments are delivered over TCP, network-level fairness between multiple streaming sessions is achieved by using congestion control strategies. However, fairness in terms of bandwidth consumption does not necessarily correspond to fairness in perceived QoE. For example, consider a scenario where two HAS clients compete for bandwidth. The first client is close to running into a buffer starvation, the second client has a well-stocked buffer. While the TCP fair-share paradigm would result in an equal bandwidth share for both clients, the total perceived QoE could strongly benefit from temporarily increasing the bandwidth share for the first client in order to avoid a buffer starvation. Therefore,

scalable coordination between multiple HAS clients is required to further improve the fairness in terms of QoE.

6.2.3 Network protocol evolutions

Alongside the developments in video streaming services, underlying technologies and protocols are evolving as well. To be able to continuously increase the delivered service quality, it is important to embrace technology changes and fully exploit the offered advantages, both at the application layer and the transport layer. For example, transport protocols such as TCP are continuously being optimized. At the application layer, HTTP/2 was standardized in 2015, providing a set of novel features that could greatly improve the efficiency of HAS delivery. For instance, HTTP/2 allows a server to *push* content to the client, allowing to deliver more data than is initially requested. Pushing video segments to the client could strongly increase the efficiency in scenarios with high Round-Trip Times (RTTs) [2].

However, besides new possibilities, new challenges arise with changing technology. Even though the HTTP/2 standard itself does not require encryption, most client implementations only support HTTP/2 over Transport Layer Security (TLS), making encryption de facto mandatory. This trend towards mandatory encryption can also be seen in novel protocols at the transport layer, such as Quick UDP Internet Connections (QUIC). While this can strongly improve the security, it introduces some big challenges when it comes to caching and in-network optimizations in general.

6.2.4 Network virtualization and SDN

In Chapter 3 it was demonstrated how virtualization of storage and networking resources can open up new business models to the ISPs by enabling them to simultaneously lease their telco-CDN infrastructure to multiple third parties. Periodical capacity reallocation and content placement have been proposed to reduce the load on the ISP network resources. Alongside network virtualization, Software-Defined Networking (SDN) has gained a lot of attention in recent years. With this networking paradigm, the control plane is decoupled from the forwarding plane, allowing delivery paths to be dynamically rerouted based on the current requirements. For example, an SDN-based approach could be applied to prioritize specific video streams in the network, for example based on the current buffer filling level. In this way, the number of video freezes could be significantly reduced. However, it is clear that routing and content placement decisions are closely coupled as there is a mutual influence between the results of both decisions. Therefore, we argue that both decisions should be considered simultaneously, significantly increasing

the problem complexity. Furthermore, in the light of Network Function Virtualization (NFV) and SDN, the work presented on dynamic cache capacity allocation can be extended to consider dynamic placement of middleboxes.

References

[1] J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. Alface, T. Bostoen, and F. De Turck. *HTTP/2-based adaptive streaming of HEVC video over 4G/LTE networks*. IEEE Communications Letters, 20(11):2177–2180, 2016.

[2] R. Huysegems, J. van der Hooft, T. Bostoen, P. Alface, S. Petrangeli, T. Wauters, and F. De Turck. *HTTP/2-based methods to improve the live experience of adaptive streaming*. In Proceedings of the ACM International Conference on Multimedia, pages 541–550, 2015.

A

Controlling the AIMD Behavior of Deadline-aware TCP Congestion Control Algorithms in HAS

In Chapter 5, a deadline-aware Transmission Control Protocol (TCP) congestion control strategy was proposed, based on a parametrization of TCP New Reno. Evaluations showed that performance gain could be achieved by dynamically changing both parameters α and β, respectively influencing the Additive Increase/Multiplicative Decrease (AIMD) of the congestion window. However, the best results are generally achieved by focusing on the parameter α. In this appendix, we extend the work presented in that chapter by introducing a novel congestion control strategy that is able to adjust both the α and β parameters simultaneously, depending on the perceived congestion state. Furthermore, while in Chapter 5 it was shown how the bottleneck link bandwidth could be reduced in a non-segment-based video streaming scenario over TCP, this appendix demonstrates the performance of both proposed algorithms in an HTTP Adaptive Streaming (HAS) scenario over access networks. It is shown that using the proposed approach, the average estimated Mean Opinion Score (eMOS) can be increased by up to 11% while reducing the total video freezing time with as much as 94%.

A.1 Introduction

In Section 5.3.1 of this dissertation, a parametrization of the Transmission Control Protocol (TCP) New Reno congestion control strategy was proposed. Introducing two parameters α and βallows to tune the modulation of the congestion window in case of a packet acknowledgment or a congestion event, respectively. Based on this parametrization, a deadline-aware congestion control strategy that dynamically adapts the value of one of both parameters, based on the available deadline information, was proposed in Section 5.4. The evaluation results presented in Figure 5.7 and Figure 5.8 showed that in general higher performance can be achieved by dynamically changing the value of α. This finding could be explained by the role of both parameters in the congestion control mechanism. While the value of α has an impact for every received acknowledgment, the value of β only impacts the behavior in the event of congestion. Therefore, the impact of dynamically adapting β on the congestion window is perceived less frequently. However, when congestion is perceived, the value β still has a significant impact on the aggressiveness of the stream. To further clarify this issue, a conceptual example is presented in Figure A.1, showing the typical sawtooth behavior of the TCP congestion window size. Assume that at time 0.4 it is detected that the throughput of the stream is too low and the aggressiveness should be increased, either by lower the value of α or by decreasing β. It can be seen that lowering α immediately results in an increased aggressiveness while initially no effect is perceived for adjusting the value of β. However, at the next congestion event, the lowered value of β results in a limited reduction of the congestion window, significantly increasing the aggressiveness of the stream. While no immediate effect could be perceived, adjusting the value of β is very efficient on the long term. Therefore, in this appendix, a modified deadline-aware congestion control strategy is proposed that estimates the likeliness of perceiving a congestion event and takes this information into account to simultaneously modify the value of α and β. It is important to emphasize again that the proposed approach only requires server-side adaptations at the TCP level and is fully transparent in the rest of the network, strongly benefiting deployability. In Section A.2, the details of this algorithm are presented. In the remainder of this appendix, the algorithm presented in Chapter 5 will be referred to as *margin-based* while the new algorithm will be denoted as *congestion-aware*. The impact of both algorithms is evaluated in terms of Quality of Experience (QoE) in an HTTP Adaptive Streaming (HAS) scenario and reported in Section A.3.

A.2 Algorithm

The most relevant notations introduced in Chapter 5 are summarized in Table A.1. The goal of the proposed congestion control algorithm is to keep the achieved

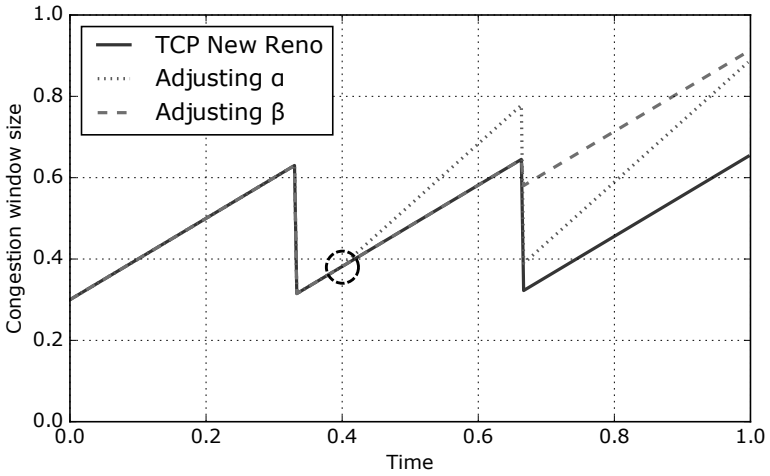


Figure A.1: Conceptual demonstration of the delayed influence of adjusting β.

Table A.1: Notation summary for deadline-aware congestion control.

Notation	Definition
s	Video stream
t	Current time
r_s	Current bit rate of stream s
l_s	Total video length in stream s
d_0	Begin deadline of a stream
$d_e = d_0 + l_s$	Last deadline of a stream
$d_b = d_0 + \frac{b*8}{r_s}$	Deadline of byte b in a stream
$m = d_b - t$	Deadline margin of byte b
m_l	Lower bound for deadline margin
m_u	Upper bound for deadline margin
α_l, β_l	Lower bound for α, β
α_u, β_u	Upper bound for α, β
α_g, β_g	Granularity of α, β adaptations
t_r	Reactivity time

throughput and the deadline margin as close as possible to the current bit rate r_s of the video stream and a predefined target margin m_t, respectively. Therefore, the algorithm will adjust the value of α based on the ratio between the current throughput and the bit rate of the stream and the difference between the current and the target deadline margin. These adaptations try to keep the throughput and deadline margin as close possible to the target values in the current congestion state. However, when the level of congestion changes, the achieved throughput is impacted. As explained in the previous section, changing the value of β can be very effective to change the aggressiveness of a stream in case of congestion. Therefore, the value of β is adjusted based on the likeliness of congestion to occur. If a congestion event is unexpected, the stream should become more aggressive to maintain the current throughput level. To be able to do so, the algorithm continuously keeps track of the average time μ_c between consecutive congestion events. When congestion is perceived at time t while the previous congestion event occurred at time t_c, μ_c is updated as an Exponentially Weighted Moving Average (EWMA) with parameter ϵ, as defined in (A.1).

$$\mu_c = \epsilon * (t - t_c) + (1 - \epsilon) * \mu_c \tag{A.1}$$

Algorithm A.1 presents the pseudo-code of the proposed algorithm. This algorithm is executed every t_r seconds, as long as no deadlines were missed for the considered stream. To avoid congestive collapse and to maintain a degree of fairness with other TCP flows, a streaming session falls back to TCP New Reno congestion control (i.e. α=1.0, β=0.5) when a deadline was missed. As a first estimate of α the throughput ratio R_t between the current throughput estimate T and the current bit rate r_s of the video is calculated (line 1). When the current throughput matches the video bit rate, this results in a default value of 1. When the current throughput is insufficient, a more aggressive value is obtained (i.e. $R_t < 1.0$). Next, based on the current time t and the deadline d_b for the next byte to send b, the current deadline margin m is calculated (line 2) and its relative deviation R_m from the target margin m_t is defined (line 3). Based on this relative deviation, the urgency factor u is calculated using a shaped sigmoid function $ssigm$ as defined in (A.2) (line 4). This urgency factor u is then used as a multiplier for R_t to calculate the final value of α (line 5). The final value is rounded to a multiple of α_g between the predefined bounds $[\alpha_l; \alpha_u]$. The rationale behind the urgency factor u is to decrease the value of α compared to R_t when the deadline margin is too low, and to increase it when the margin is too high. As shown in Figure A.2, the exact shape of the function $ssigm$, used to calculate the urgency factor, depends on the shape factor σ, while its range is limited to $[0.5; 2.0]$.

$$\text{ssigm}(x, \sigma) = \frac{1.5}{1 + \exp(-1 * \sigma * (x - \frac{\ln(2)}{\sigma}))} + 0.5 \tag{A.2}$$

Input:

t: current time
T: current throughput measurement
d_b: deadline of next byte to send
r_s: current bit rate of the stream
t_c: last congestion event
μ_c: average time between consecutive congestion events

1: $R_t = \frac{T}{r_s}$
2: $m = d_b - t$
3: $R_m = \frac{m - m_t}{m_t}$
4: $u = \text{ssigm}(R_m, \sigma)$
5: $\alpha = \text{roundBetweenBounds}_{\alpha}(R_t * u)$
6: $\beta = \text{roundBetweenBounds}_{\beta}(\frac{t - t_c}{2 * \mu_c})$

Algorithm A.1: Outline of the proposed algorithm to dynamically adapt the parameter value α and β based on the current deadline margin and congestion state. This update is performed every t_r seconds.

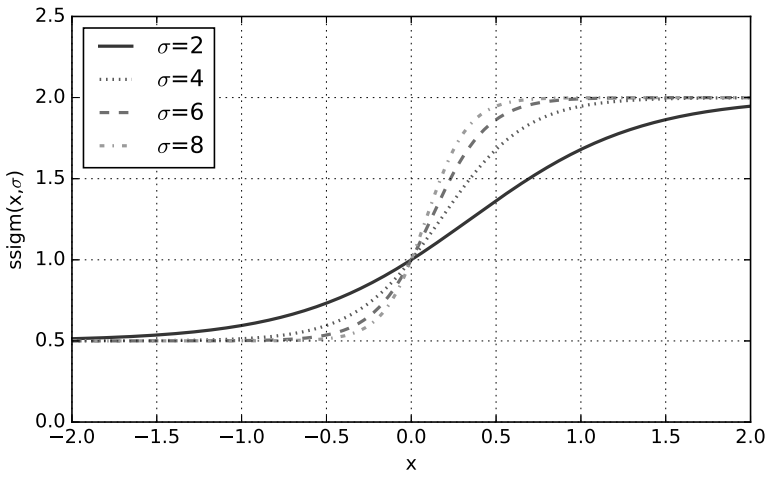


Figure A.2: Example of the shaped sigmoid function $ssigm$, used to calculate the urgency factor, for 4 different values of σ.

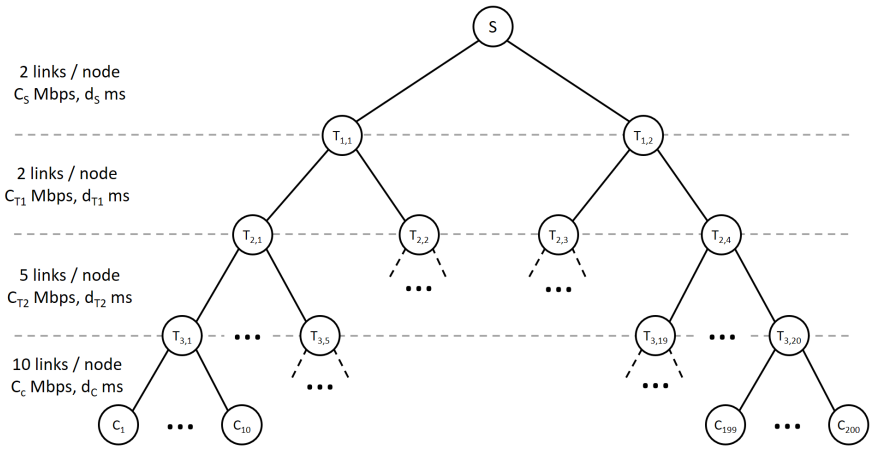


Figure A.3: Evaluated tree-based topology.

Finally, the current time since the last congestion event (i.e. $t - t_c$) is compared to the average time between consecutive congestion events. When the current time is above average, the level of congestion is decreasing and more back-off can be afforded in case of congestion (i.e. $\beta > 0.5$). When the level of congestion is increasing, the current stream should back-off less to be able to maintain its current throughput level (i.e. $\beta < 0.5$). Based on this rationale, the value of β is calculated as $\frac{t-t_c}{2*\mu_c}$. The final value is again rounded to a multiple of β_g between the predefined bounds $[\beta_l; \beta_u]$ (line 6).

A.3 Scenario description

To evaluate the *congestion-aware* and *margin-based* deadline-aware congestion control strategies, a typical tree-based network topology is considered. This topology consists of 1 HAS server S, 2 distribution nodes $T_{1,x}$, 4 aggregation nodes $T_{2,x}$, 20 delivery nodes $T_{3,x}$ and 200 client nodes C_x. The structure of this topology is presented in Figure A.3. Two topologies, denoted as tree 1 and tree 2, with different bottleneck locations are considered. The different link capacities C and delays d in both topologies are listed in Table A.2, where the bottleneck location is marked with ✓.

Over these topologies, multiple Video-on-Demand (VoD) sessions are streamed. The number of active clients is varied throughout the evaluations, with activation levels ranging from 20% to 80% of active clients. All sessions have a fixed duration of 30min, streaming a video available in 4 bit rate representations (560kbps, 1050kbps, 2350kbps, 4300kbps), based on the quality representations provided by

Table A.2: Notation summary for deadline-aware congestion control.

		Tree 1		Tree 2
C_S		200Mbps		200Mbps
d_S		20ms		20ms
C_{T1}		100Mbps	✓	100Mbps
d_{T1}		10ms		10ms
C_{T2}	✓	20Mbps		40Mbps
d_{T2}		5ms		5ms
C_C		10Mbps		10Mbps
d_C		2ms		2ms

Table A.3: Considered parameter configurations for the *margin-based* algorithm.

Parameter	**Values**
Adjusted parameter	α, β
m_l-m_u	0-10s, 0-20s, 5-10s, 5-20s, 10-20s

Netflix[1]. A fixed segment duration of 2s is assumed, as is commonly the case in Dynamic Adaptive Streaming over HTTP (DASH). The start times of the different clients are uniformly distributed over a period of 30min, while the video playout is started 5s after the start of the stream. All clients use the FINEAS rate adaptation heuristic [1]. These assumptions allow us to analyze the results in terms of level of link contention. Under these conditions, with an activation level of $x\%$, $x\%$ of the clients will be active simultaneously at the peak moment, while 0.5*$x\%$ of the clients will be active simultaneously on average. When variable session durations would be considered, no conclusions can be drawn about the peak level utilization and the corresponding link contention. The size of the buffer available at the client side is varied between 20s, 60s and 600s. Due to the randomization in the session start times, for each configuration, 5 different request traces are generated. This results in a total of 120 evaluation scenarios (3 buffer sizes * 4 activation levels * 2 topologies * 5 request traces). All of these scenarios are simulated in NS-3 using TCP New Reno as well as both of the proposed congestion control algorithms. The considered parameter configurations for the *margin-based* and *congestion-aware* algorithms are presented in Table A.3 and Table A.4, respectively.

To assess the quality of the proposed approaches, the QoE is considered as a performance metric. In literature, it is commonly assumed that the factors influencing the QoE of a HAS service are the average quality level, the quality switching behavior and the video freezes [2, 3]. Therefore, for each client the average played quality level, its standard deviation, the number of video freezes and the total time of the video freezes is calculated. Based on these values, an estimated Mean Opin-

[1]Netflix Tech Blog - http://techblog.netflix.com/2015/12/per-title-encode-optimization.html

Table A.4: Considered parameter configurations for the *congestion-aware* algorithm.

Parameter	Values
ϵ	0.3
m_t	5s, 10s, 20s
σ	2.0, 4.0, 6.0, 8.0

ion Score (eMOS) can be calculated using equation (2.11), presented in Chapter 2. This model has been slightly redefined to consider an arbitrary number of available quality levels. The resulting formula can be found in (A.3), where μ and σ represent the normalized average quality level and its standard deviation, respectively [4]. The value of ϕ can be calculated using equation (2.10) and estimates the impact of the video freezes. To evaluate the achieved gain compared to standard TCP congestion control, all results will be presented relatively to the performance of TCP New Reno in the same scenario, unless stated differently.

$$\text{eMOS} = \max\left(5.67 * \mu - 6.72 * \sigma - 4.95 * \phi + 0.17, 0\right) \tag{A.3}$$

A.4 Evaluation results

To keep the analysis of the results of the wide range of simulations manageable, in Section A.4.1, the influence of the algorithm parameters is evaluated, averaged over all scenarios. Next, using the selected optimal parameter configurations, the influence of the scenario characteristics is investigated in Section A.4.2.

A.4.1 Parameter analysis

To identify the best parameter configuration of both algorithms, we compare the performance of the *margin-based* and *congestion-aware* algorithm to the performance of TCP New Reno in terms of eMOS. To be able to manage the analysis, results are averaged over the different scenarios (topology, activation interval, buffer size, request trace). As a result, the presented values are averaged over 120 scenarios.

Figure A.4 shows the average relative eMOS for the different parameter configurations of the *margin-based* algorithm. It can be seen that the best performance is obtained for higher deadline margin bounds (i.e. 5-20s and 10-20s). This matches the findings presented in Chapter 5. When the margin bounds are set closer to the deadline, the algorithm has less time to react, resulting in a slight performance degradation. Furthermore, it can be seen that, as opposed to the results presented in Chapter 5, the best performance is achieved when dynamically changing the value of β. This seeming mismatch of results can be attributed to the fact that Chapter 5

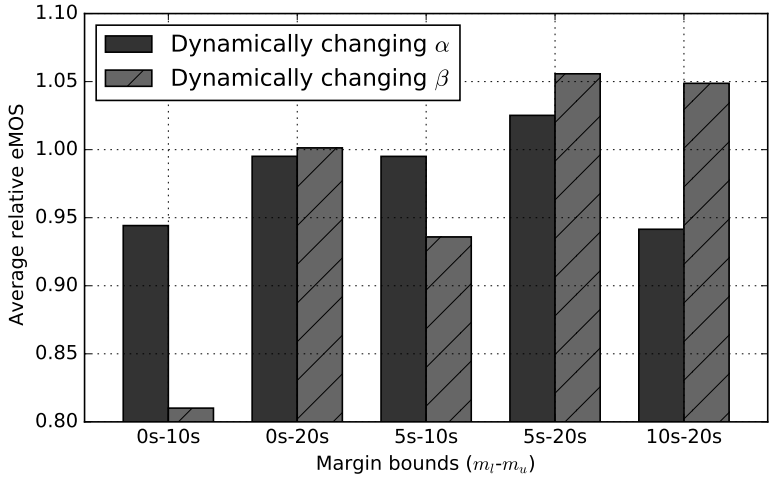


Figure A.4: Average performance of the *margin-based* congestion control algorithm for different parameter configurations.

focused on bottleneck bandwidth reduction to allow video streaming without interruptions. In the current evaluations, the best results in terms of video freezing are obtained by adapting the value of α as well. However, the performance in terms of average quality level and switching behavior is significantly higher when considering the β parameter, resulting in a higher eMOS. Based on these evaluations, dynamically changing the value of β with deadline margin bounds configured to m_l=5s and m_u=20s generally yields the best performance for the *margin-based* algorithm.

Similarly, the average relative eMOS for the different parameter configurations of the *congestion-aware* algorithm is presented in Figure A.5. It can immediately be seen that the performance of this algorithm is less sensitive to the specific parameter configuration compared to the *margin-based* algorithm, with an average performance increase of more than 5% for all configurations. However, it can also be seen that on average, the best performance is obtained with a target margin of m_t=10s. When a lower target margin is used, the algorithm has less time to react to approaching deadline misses. On the other hand, with higher target margins, a flow will back-off later, leaving less bandwidth for other flows. In terms of the sigmoid shape factor, a value of σ=6.0 appears to be the best trade-off on average. Based on these findings, the average optimal parameter settings for the *congestion-aware* algorithm are chosen as m_t=10s and σ=6.0.

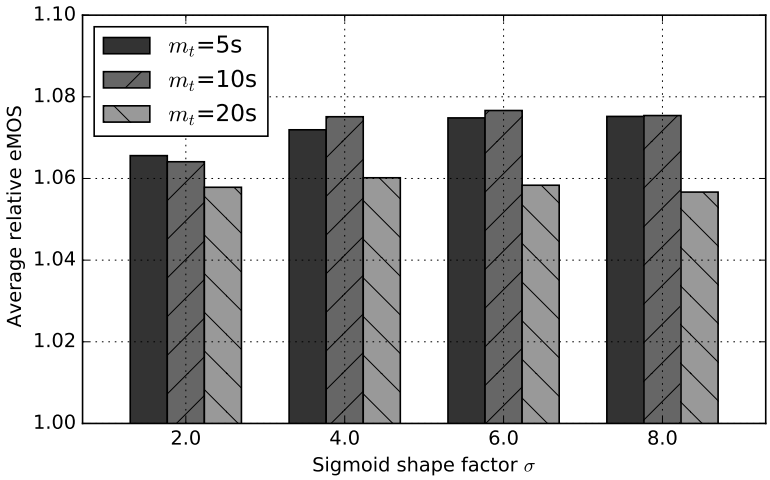


Figure A.5: Average performance of the *congestion-aware* congestion control algorithm for different parameter configurations.

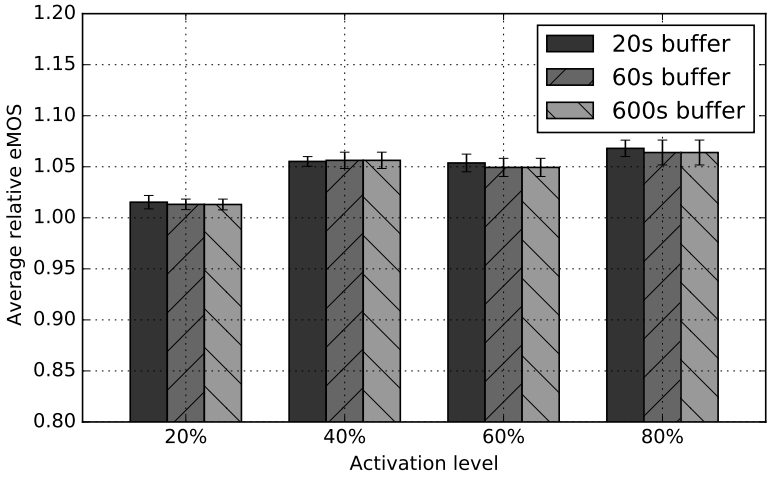


Figure A.6: Average performance of the *margin-based* congestion control algorithm using optimal parameter configurations in tree topology 1 for multiple activation levels and client buffer sizes.

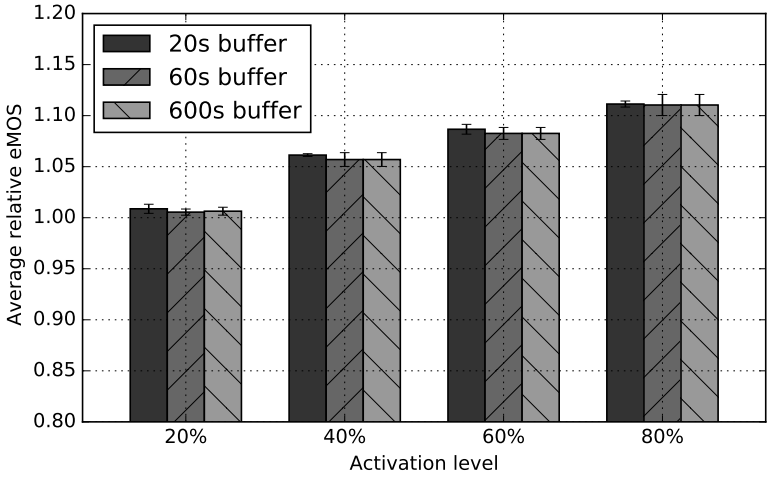


Figure A.7: Average performance of the *margin-based* congestion control algorithm using optimal parameter configurations in tree topology 2 for multiple activation levels and client buffer sizes.

A.4.2 Influence of scenario characteristics

Based on the optimal parameter configurations described above, in this section, the influence of the scenario characteristics on the performance of both algorithms is evaluated. Figure A.6 and Figure A.7 show the average relative performance and its standard deviation in terms of eMOS for the *margin-based* algorithm in different scenarios, using topology tree 1 and tree 2, respectively. For each scenario, the results are averaged over the 5 request traces to cover the randomness in the session start times. It immediately becomes clear that for both topologies and for all activation levels, the influence of the client buffer size is negligible. This can be explained by the fact that we are dealing with congested scenarios. Using HAS, only in over-provisioned scenarios where a client is able to continuously stream the highest quality, significant buffers can be built up. However, these results show that using the proposed congestion control algorithm, fairness is maintained between the different clients. When a video client starts earlier and switches to a higher quality level, one could expect that trying to maintain the current throughput would limit the bandwidth available for newly starting video streams, resulting in a lower average eMOS. However, by taking into account the deadline margin to back-off when the buffer filling becomes too large, this is not the case.

Furthermore, even though it is most clear for topology tree 2, it can be seen that for both topologies, the performance gain grows with the activation level. When more video streaming sessions are simultaneously active, the level of congestion increases. In this case, more benefit can be achieved by implicit coordination be-

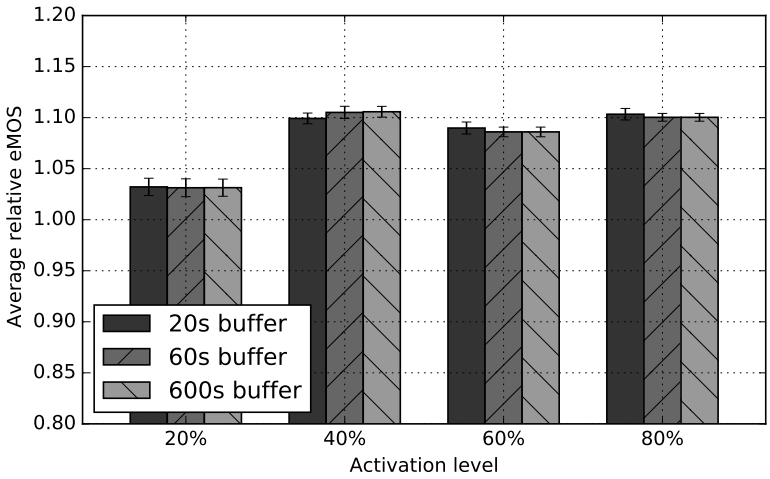


Figure A.8: Average performance of the *congestion-aware* congestion control algorithm using optimal parameter configurations in tree topology 1 for multiple activation levels and client buffer sizes.

tween the different streams, as is the case in the proposed approach. This also explains why, for high activation levels, higher performance is obtained in topology tree 2. As can be seen in Table A.2, in this topology, the bottleneck links are located higher in the tree, causing more streaming sessions to share a common bottleneck link. This again allows for more benefit by implicit stream coordination.

In a scenario with high activation levels (i.e. 80%) in topology tree 1, using the proposed *margin-based* congestion control algorithm results in a slight increase of 1.73% in terms of average quality level and a decrease of 2.48% in terms of its standard deviation, compared to TCP New Reno. Furthermore, the total number of freezes is reduced by 22.11%, while the total freeze time is reduced with 25.09%. Combining these factors results in an increase of 6.55% in terms of eMOS. In topology tree 2, the average performance gains in terms of average quality level, its standard deviation, the total number of freezes and the total freeze time amount to 2.95%, 4.63%, 47.98% and 55.92%, respectively. This results in an average eMOS increase of 11.07%.

Similarly, Figure A.8 and Figure A.9 show the performance of the *congestion-aware* algorithm in both topologies. In terms of the client buffer size, the same conclusions can be drawn as for the *margin-based* algorithm. Furthermore, it can be seen that, as opposed to the *margin-based* algorithm, the difference in performance gain between medium and high activation levels is limited in both topologies. This can be explained by the fact that this algorithm adapts the value of both the α and β parameters simultaneously, while the *margin-based* algorithm focuses

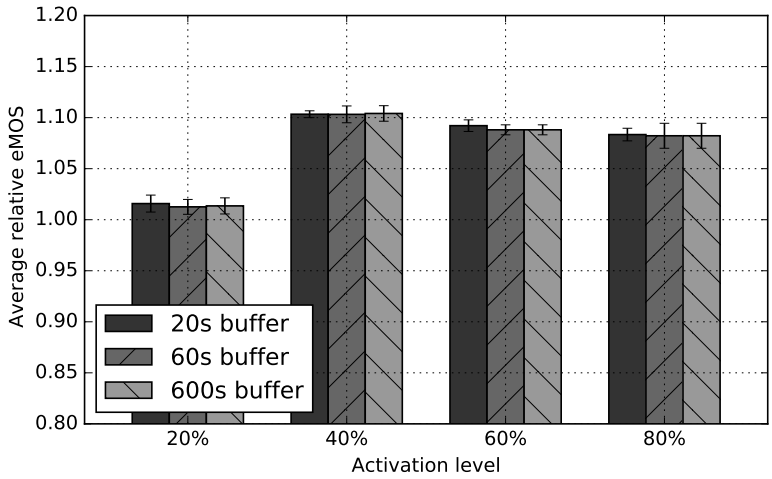


Figure A.9: Average performance of the *congestion-aware* congestion control algorithm using optimal parameter configurations in tree topology 2 for multiple activation levels and client buffer sizes.

on the value of β only. When there is limited congestion, the value of β only has a limited impact, restricting the performance gain for the *margin-based* algorithm. However, in these scenarios with limited congestion, the *congestion-aware* algorithm can already benefit from the adaptation of α, which has a more fine-grained impact frequency. In the scenarios with low activation levels (i.e. 20%), the congestion level is negligible, limiting the possibility of performance gain for both algorithms. It can be seen that the *congestion-aware* algorithm always outperforms the *margin-based* algorithm, except in a scenario with high activation levels and a high number of clients sharing a common bottleneck link (i.e. topology tree 2 and 80% activation level). In this scenario, combining adaptations of α and β yields a slight increase in the number of congestion events, resulting in a limited performance reduction compared to the *margin-based* algorithm.

With an increase of 0.10%, using the proposed *congestion-aware* congestion control algorithm has an negligible influence on the average quality level in a scenario with high activation levels (i.e. 80%) in topology tree 1. However, the standard deviation of the quality level is reduced with 3.51% on average. Most importantly, the total number of video freezes is reduced by as much as 92.65% while the total freeze time is decreased with 94.34%. Combining these factors yields an increase of 10.13% in terms of eMOS. In the same scenario over topology tree 2, the average performance gains amount to 0.42%, 2.68%, 91.62%, 93.72% and 8.26% in terms of average quality level, its standard deviation, total number of freezes, total freeze time and eMOS respectively.

Table A.5: Average performance gain of the *margin-based* (MB) and *congestion-aware* (CA) algorithm compared to TCP New Reno in a scenario with high activation levels of 80%.

		QL	**Switches**	**# freezes**	**Freeze time**	**eMOS**
Tree 1	MB	1.73%	2.48%	22.11%	25.09%	6.55%
	CA	0.10%	3.51%	92.65%	94.34%	10.13%
Tree 2	MB	2.95%	4.63%	47.98%	55.92%	11.07%
	CA	0.42%	2.68%	91.62%	93.72%	8.26%

The performance gain in scenarios with high activation levels is summarized in Table A.5. It can be seen that the *congestion-aware* algorithm is able to achieve a significantly higher video freeze reduction, while having limited impact on the displayed quality level. For the *margin-based* algorithm, the gain in terms of video freezes is more limited, but the increase in terms of eMOS is compensated by a higher average quality level.

A.5 Conclusions

This appendix has built on the work presented in Chapter 5 by introducing a deadline-aware congestion control strategy that dynamically adapts both parameters of the parametrized TCP New Reno algorithm. This algorithm takes into account the role of both parameters in the modulation of the congestion window size to optimize the adaptations to the perceived congestion state. Furthermore, while the evaluations presented in Chapter 5 focused on reducing the bottleneck link bandwidth, new evaluations have been presented in this appendix, targeting the optimization of the QoE. It was shown that, depending on the level of congestion, the average eMOS was increased by up to 11%. Furthermore, by using the proposed approach, the total video freezing time was reduced by as much as 94%. These results show that by dynamically adjusting the modulation of the congestion window, the number of deadline misses can significantly be reduced without impacting the average video quality.

References

[1] S. Petrangeli, J. Famaey, M. Claeys, S. Latré, and F. De Turck. *QoE-driven rate adaptation heuristic for fair adaptive video streaming*. ACM Transactions on Multimedia Computing, Communications, and Applications, 12(2):1–24, 2015.

[2] J. De Vriendt, D. De Vleeschauwer, and D. Robinson. *Model for estimating QoE of video delivered using HTTP adaptive streaming*. In Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM), pages 1288–1293, 2013.

[3] R. Mok, E. Chan, and R. Chang. *Measuring the quality of experience of HTTP video streaming*. In Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM), pages 485–492, 2011.

[4] M. Claeys, S. Latré, J. Famaey, and F. De Turck. *Design and evaluation of a self-learning HTTP adaptive video streaming client*. IEEE Communications Letters, 18(4):716–719, 2014.