Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática
2010

Daniel Filipe Leonardo Figueira

Controlo do Transporte de Sessões Multicast em Redes Dinâmicas

Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática
2010

Daniel Filipe Leonardo Figueira

Controlo do Transporte de Sessões Multicast em Redes Dinâmicas

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e de Telecomunicações, realizada sob a orientação científica da Prof. Dra. Susana Sargento, Professora auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro e do Prof. Dr. Augusto Neto, Colaborador do Instituto de Telecomunicações e Professor Adjunto da Universidade Federal de Goiás.

Dedico este trabalho aos meus Pais, Irmã e Namorada pelo incansável apoio.

O júri

Presidente	Prof. Dr. José Alberto Gouveia Fonseca Professor associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.
Orientadora	Prof. Dra. Susana Isabel Barreto de Miranda Sargento Professora auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.
Vogal	Prof. Dr. Fernando José Silva Velez Professor auxiliar do Departamento de Engenharia Electromecânica da Faculdade de Engenharia da Universidade da Beira Interior.

Agradecimentos

Desde logo gostaria de agradecer de forma especial aos meus Pais, Irmã e Namorada, pelo constante apoio, atenção e motivação para superar esta importante etapa da minha vida.

À professora Susana Sargento, por toda a disponibilidade, orientação e motivação ao longo de todos estes meses, imprescindível para a conclusão com sucesso deste trabalho.

Ao Nuno Carapeto por ter partilhado comigo todos os seus conhecimentos, estando sempre disponível para tirar duvidas para a elaboração do trabalho.

Ainda a todos os meus colegas de curso que tanto me apoiaram, tanto nos melhores momentos como nos mais complicados, um grande obrigado.

Palavras-chave

Multicast, Consciencia de Contexto, Sessoes Multiparty, Qualidade de Experiencia, Qualidade de Serviço.

Resumo

Hoje em dia assiste-se a um aumento considerável da procura de serviços ou aplicações para múltiplos utilizadores (*multiparty*), como *streaming* de conteúdos *media*, partilha de informações, colaboração entre utilizadores, entre outros. O suporte de tais serviços pela Internet mostra-se extremamente exigente para as actuais arquitecturas de rede, requerendo recursos e funcionalidades completamente fora do alcance de serem suportadas. Desta forma, é necessário investigar e desenvolver novos mecanismos que possibilitem simultaneamente um melhor e maior controlo a este nível.

Neste âmbito, tem-se observado actualmente um aumento no número de arquitecturas propostas, capazes de integrar as vantagens do protocolo IP multicast na entrega e transporte de conteúdos multimédia a grupos de utilizadores. Por outro lado, a inclusão de informação de contexto da rede, ambiente e utilizadores proporciona uma maior personalização e adaptação nas decisões de controlo necessárias à rede. É neste sentido que o projecto C-Cast abordado nesta tese se enquadra, procurando especificar uma arquitectura capaz de integrar uma quantidade abrangente de informação de contexto por forma a fornecer personalização nas sessões de entrega de conteúdos multimédia. Para a gestão dos recursos de rede é também proposto um mecanismo de transporte dos conteúdos baseado em IP multicast, juntamente com a possibilidade de adaptação dos caminhos escolhidos na rede de *core* baseada na informação de contexto da rede e do utilizador. A junção destes factores visa então possibilitar uma melhor gestão dos recursos disponibilizados pela rede. O principal objectivo desta Tese é então focado no desenvolvimento de um módulo inteligente, capaz de permitir o transporte *multiparty* e reserva de recursos, permitindo a entrega dos conteúdos de uma forma personalizada e independente das capacidades da rede e do utilizador, melhorando não só a qualidade de serviço, como também a qualidade de experiência.

De forma a implementar a proposta apresentada, recorreu-se à criação de um novo componente organizado segundo uma arquitectura interna hierarquizada e centralizada, na qual um único módulo (IPT Controller) central comanda vários outros módulos (IPT Node) distribuídos ao longo de toda a rede. Após concluída a implementação, provou-se que o componente é capaz de criar e remover várias reservas de recursos por forma a permitir o transporte *multiparty* por caminhos específicos na rede. Este mecanismo é também capaz de modificar reservas previamente efectuadas, permitindo que as sessões multimédia sejam capazes de responder às modificações de contexto na rede, actualizando possíveis sessões já existentes com um mínimo de interrupção de serviço possível para utilizadores que não tenham sofrido alterações. Avaliando os resultados obtidos, pode-se também concluir que o impacto do IPT na rede é ligeiro e menor em relação ao dos restantes componentes, pelo que não é um factor decisivo no desempenho global da arquitectura.

Keywords

Multicast, Context-Awaress, Multiparty Sessions, Quality of Experience, Quality of Service.

Abstract

Today we are witnessing a considerable increase in the demand for services or applications for multiple users (multiparty), such as streaming media content, information sharing, collaboration among users, among others. The support of such services over the Internet proves to be extremely demanding for the existing network architectures, requiring features and functionality completely out of reach of the current networks. Thus, investigation and development of new mechanisms that enable a superior management at this level is necessary.

In this context, it is possible to observe an increase in the number of architectures proposed, which are able of integrating the advantages of the IP multicast protocol in the transport and delivery of multimedia content to user groups. Moreover, the inclusion of context information from the network, environment and users, provides a greater customization and adaptation in the decisions necessary to control the network. It in this context that the C-CAST project discussed in this dissertation is included, which tries to specify an architecture capable of integrating a comprehensive amount of context information in order to be able to provide superior dynamic sessions to deliver multimedia content. For the management of the network resources, it is also proposed a mechanism for the transport of multimedia content based on IP multicast. The possibility of path adaptation in the core network based on context information of both users and network is also considered. Is it with the combination of these factors that this project seeks to enable an improved management of the network's resources. The main objective of this dissertation is then focused on the latter point of the C-CAST project architecture. It proposed the development of an intelligent module capable of allowing multiparty transport and resource reservation, enabling the delivery of multimedia content in a personalized way, independent of network and users capabilities, improving not only the quality of service, as well as quality of experience.

In order to implement the proposed solution, a new component was developed, organized according to a hierarchical and centralized architecture in which a single central unit (IPT Controller) is able to command several other modules (IPT Node) deployed throughout the network. It was verified that the developed component is capable of creating and removing the necessary enforcements on the network to enable the multiparty transport through specific data paths. It is also capable of modifying previous enforcements, allowing the multimedia sessions to adapt themselves to context changes avoiding as much as possible to disrupt the existing services to users that were not subjected to modifications. Evaluating the obtained results, it is possible to conclude that the overall impact of the IPT component in the network is reduced and considerably less than the one of the remaining components. Hence, the IPT does not have a direct impact on the overall architecture performance.

Table of Contents

List of Figures

Index of Tables

Acronyms

A

AAA	Authentication, Authorization, and Accounting
ABC	Always Best Connected
ALM	Application Layer Multicast
AMT	Abstract Multiparty Transport
AVP	Attribute-value Pair

B

BT	Bi-Directional Tunnelling

C

CBT	Core Based Tree
C-CAST	Context Casting Project
CDE	Content Delivery Enabler
CoS	Class of Service
CSE	Content Selection Enabler
CtPD	Content Processing and Delivery
CxB	Context Broker
CxC	Context Consumer
CxH	Context History
CxP	Context Provider

D

DVMRP	Distance-Vector Multicast Routing Protocol

F

FA	Foreign Agent

G

GME	Group Management Enabler

H

HA	Home Agent

I

IGMP	Internet Group Management Protocol
IP	Internet Protocol
IPT	IP Transport
ISP	Internet Service Provider

M

MA	Multicast Agent
MIRA	Multi-service Resource Allocation
MN	Mobile Node
MOSPF	Multicast OSPF
MRIB	Multicast Routing Information Base
MTO	Multiparty Transport Overlay

N

NGN	Next Generation Networks
NME	Network Management Enabler
NRS	Neglected Reservation Sub-tree
NSIS	Next Steps in Signalling
NSLP	NSIS Signalling Layer Protocol
NTLP	NSIS Transport Layer Protocol
NUM	Network Use Management

O

OSMAR	Overlay for Source-Specific Multicast in Asymmetric Routing
ON	Overlay Node

P

PIM	Protocol Independent Multicast

Q

QoE	Quality of Experience
QoS	Quality of Service

R

RADIUS	Remote Authentication Dial-In User Service
RS	Remote Subscription

S

SIP	Session Initiation Protocol
SLS	Service Level Specifications
SME	Session Management Enabler
SUM	Session Use Enabler

T

TCP	Transmission Control Protocol

U

UDP	User Datagram Protocol

V

VoIP	Voice over IP

1. Introduction

1.1. Motivation

Currently, Internet Service Providers are strongly investing on new and innovative types of services, while clients are becoming increasingly more demanding. This trend is pushing actual network architectures to the limits, which are poorly prepared to support the delivery of multimedia contents to several groups of users simultaneously. Therefore, several studies and researches are being done to provide mechanisms to efficiently support the new increased requirements.

Furthermore, with the company's massive investments on mobile devices like netbooks, laptops and smartphones and their enormous success on the general population, mobility is increasingly becoming a huge factor for Next Generation Networks (NGN). Hence, clients now expect that the new services offered by the ISPs are able to be delivered to them anywhere in the world, at any time.

A good solution to relieve some of the new requirements imposed in the networks is the IP multicast. Being group oriented, the IP multicast is ideal for the transmission of the multimedia contents to the groups of users, since it is able to grant a significantly better management of the network resources. However, due to the small support along the existing routers in the Internet, supplemented with the weak solutions for mobility, IP multicast is not yet a reliable alternative.

On the other hand, over the last years, several new access technologies as ADSL, Wi-Fi, UMTS, GPRS, WiMAX and 3G were deployed in the network, significantly increasing the range of possible underlying technologies for user's terminals to connect to. Hence, NGNs are required to fully support data transmission to connected terminals independently of the access technology used.

Considering this new framework, new solutions appeared with the purpose of addressing the current network limitations. Context-aware networks have recently gained more popularity, since several context-information can be retrieved from the network, environment and user's terminals, providing essential information which can be used to deliver improved services to the clients. It is particularly useful for the Always Best Connected (ABC) concept, since the context-information can be used to select the best possible access for each client, considering not only the network resources, but also user preferences and environment conditions. Context information can also be used in the core network, efficiently providing information to select the best path to the data transmission, but also to successfully and correctly update the network forwarding paths in situations where the network has been changed (e.g. link failure, degraded wireless signal, noisy environment, etc).

Therefore, context-awareness is a possible solution to improve the current multimedia services. However, several mechanisms have to be developed to guarantee the expected QoS. Context-information should be used to join users in groups with similar context (i.e. preferences, capabilities, requirements). IP multicast should also be used to provide the multimedia content delivery to user groups, but also considering QoS requirements. As such, access and core network paths through which the multimedia data is sent should be carefully selected, not taking only into account QoS parameters, but also the user and network context information, improving at the same time not only the QoS experienced, but also the QoE provided.

Following these thoughts, a new approach was proposed, in the framework of ICT C-Cast project [1], capable of allowing personalized session content delivery to multiple users with guaranteed resources. The delivery of the multimedia contents should also, as proposed, be independent of the underlying network access and transport technologies. It should also take into account the several context data able to be retrieved from the network, users and the surrounding environment, to maximize the delivery of the multimedia contents.

Three distinct frameworks can be identified for the C-Cast project: the context detection and delivery framework, focusing on the necessary mechanisms to gather the various context information available in the network/users/environment and provide it to the context consumers; the multiparty session management framework, which emphases the various interactions necessary to manage the multimedia sessions; the multiparty transport framework, which focus on the necessary measures to ensure the correct enforcements on the network to enable the delivery of the multimedia contents to the user's terminals. The work developed in this Dissertation is then integrated in the C-Cast project, more specifically in the multiparty transport framework, in which the development of a new component responsible for the enforcement of various network setups is proposed.

1.2. Objectives

The main objective of this Msc. Dissertation is the specification, development, test and integration in the overall architecture of one of the components included in the overall C-CAST project architecture, the IP Transport (IPT).

The developed component is integrated in the C-Cast project architecture. It is able to interact with other elements in the architecture in order to provide a reliable enforcement of the multiparty path trees on the network. It also enables support for the modification of existing sessions, in order to provide a mechanism to keep and adapt the network to possible context changes. These modifications should be able to update the network with minimal service disruption for existing users that were not subjected to modifications. Furthermore, it is also able to completely manage (create, maintain and remove) all the multicast groups in the multimedia sessions. Finally, the IP multicast addresses are also assigned by the IPT, requiring this module to also behave as a simple DHCP.

The proposed solution was subjected to various test scenarios (both alone and integrated with the remaining architecture) in order to assure the correct operation of the developed component. It is then thoroughly tested in order to evaluate its functionalities and the impact that the IPT component has on the network performance, analysing factors such as the control messages overhead, processing time required and scalability.

1.3. Contribution of this work

The entire solution implemented in this Dissertation is part of the C-Cast project, being included in its architecture as an essential component in the core network decision process. It enables the creation and removal of generic multimedia sessions, as well as the support for dynamic modifications to already existing sessions. The enforcement of these multimedia path trees in the network is done through another implemented component, the IPT Node module, as well as through an additional element of the C-Cast architecture, the Multiparty Transport Overlay (MTO).

During the development of the proposed component, it was subjected to various test scenarios with the remaining architecture components of the C-Cast project. Features such as the creation, modification and removal of multimedia sessions were thoroughly tested, as well as the capability to enforce and subsequently manage the multicast groups on the network, enabling multicast transport in specific data paths. On its final stage, it was verified that the IPT has fulfilled the proposed objectives specified at the beginning of this work. Thus, it was included in the demonstrator for the C-Cast project as an essential core network component, enabling the evaluation of the overall project performance and the impact that the usage of context information has in the improvement for the delivery of multimedia contents to multiparty groups of clients.

A scientific paper was recently submitted to the IEEE Globecom 2010 Workshop on Pervasive Group Communications, 'Pervasive Multiparty Transport Framework for Ubiquitous Multimedia Services' with the results of the testbed implementation.

1.4. Organization of the Dissertation

The work developed in this dissertation is organized in six main chapters.

The current chapter introduces the dissertation in the current situation of the next generation networks, presenting the goals and ambitions and contributions of this Dissertation.

The second chapter presents the current related work developments in the related areas, focusing on one essential aspect of this work, IP multicast. Moreover, QoS mechanisms, context-aware networks and signalling protocols are also studied.

In the third chapter, a deep description of the overall C-CAST project architecture is presented, defining its main components and their interactions.

The fourth chapter completely explains the component developed, explaining its internal processing, as well as the mechanisms used to develop the interfaces with the other elements in the C-CAST architecture.

The fifth chapter starts by presenting an overview of the development stages and the test scenarios used to assure the correct implementation of the proposed component. Next, a sample session is studied, presenting the interactions between the various components, where special detail is given to the IPT and the signalling messages exchanged. Finally, a performance evaluation of the developed component is executed, studying parameters such as the signalling overhead introduced on the network and the required processing time in comparison with the remaining components.

Finally, in the sixth chapter, the conclusions for the work developed during this dissertation are presented, as well as possible improvements that should be done in future to the IPT component and the overall architecture.

2. State of the Art

2.1. Organization

The current increase on the search for more and new multimedia services, capable of delivering personalized media content imposes a completely new set of requirements, which current network models are not able to fulfil.

To improve the delivery of the desired media contents, IP multicast is very promising, due to its superior resource management for group transfers. However, the lack of support of QoS and client and/or server mobility, as well as a low support by the routers through the Internet, call for the development of new architectures and mechanisms, capable of integrating not only user-context information, but network and environment context as well, to provide the delivery of these new services in dynamic sessions. As such, a significant study of the current technologies is fundamental.

Section 2.2 presents an overview of the current IP multicast protocols that support its functionalities, as well as a description of possible solutions for the lack of QoS and mobility support. It is also given an overview of the Multicast Overlay Routing, where its advantages and disadvantages in comparison to the IP multicast protocols are presented.

Section 2.3 describes an overview of context-aware and cognitive networks.

Since the work developed in this dissertation heavily depends upon the exchange of control messages between several elements in the network, signalling protocols are also studied in section 2.4. In this section, it is given a special focus to the Diameter protocol, since it is the one used in the developed work.

2.2. IP multicast

Regarding data transfer between various terminals over the Internet, there are four main approaches available; unicast, multicast, broadcast and anycast. However, this section will be focusing on unicast and multicast.

Unicast (Figure 1) transmission mode is much more oriented to point-to-point connections, meaning that if the same source is transferring data to several clients, each client has a separate connection, and as such, one copy of the same data has to be sent to each client. To transport the data, unicast can use either TCP [2] for the delivery guaranties, or UDP [3] for fast data delivery. In unicast, it is required that the source specifies the IP address of each client, allowing the routers to know where the data packets should be forwarded.

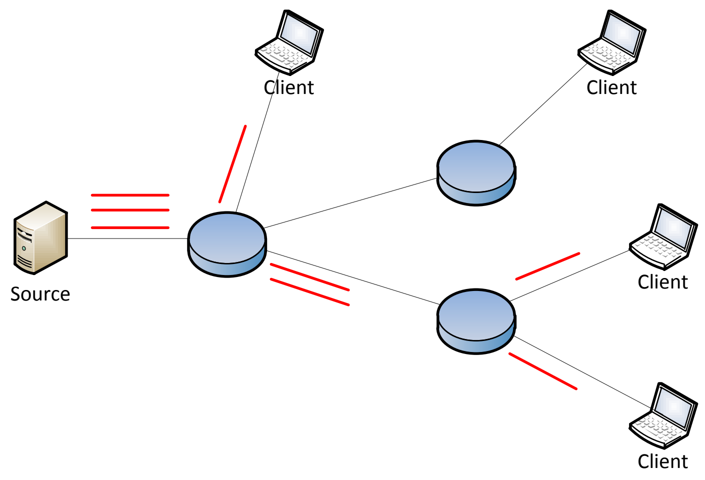


Figure 1 – Unicast approach

Multicast (Figure 2), on the other hand, is a group oriented delivery approach, without the notion of a connection between the server and the clients. Instead, the source sends the data packets to a unique IP multicast address, which represents the entire group, and as such, only one copy of the data packets is sent by the source. The forwarding of these packets is then assured by the network routers, duplicating the data packets as needed, in order to deliver them to the clients that previously joined the multicast group. To correctly setup the routing tree and forwarding rules for the multicast traffic, the IP multicast protocol uses the information from the Multicast Routing Information Base (MRIB) table, which is responsible for providing the next-hop along a multicast-capable path. Unlike the unicast protocol, IP multicast only enables the data transport through the UDP protocol. This is due to the fact that the TCP protocol forces an acknowledge message to be sent each time the client successfully receives data, which in the concept of a multicast group would imply that several acknowledge messages would be received by the source for each data packet, severely damaging the scalability of the IP multicast.

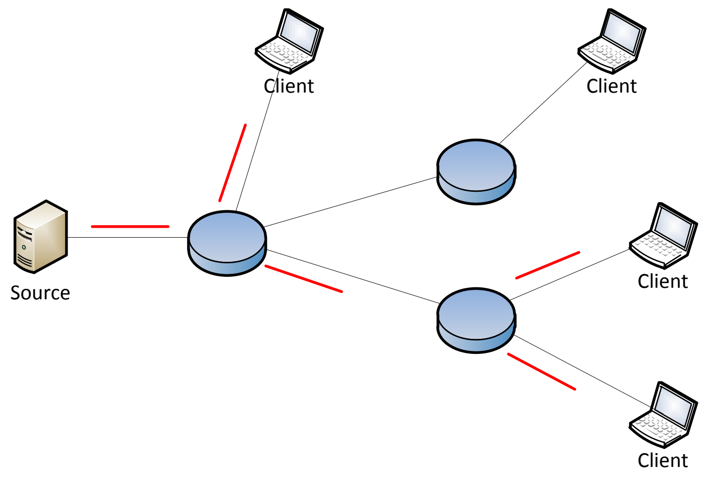


Figure 2 – Multicast approach

Packet routing in IP multicast is implemented by protocols such as Protocol Independent Multicast (PIM) [4] [5], Distance-Vector Multicast Routing Protocol (DVMRP) [6], Multicast Open Shortest Path First (MOSPF) [7], Core Based Tree (CBT) [8], among others. PIM, being the most used, presents two different versions, Dense Mode (DM) and Sparse Mode (SM). The first one assumes a scenario in which all clients are considered as interested in the multicast group, unless they specifically stated otherwise. Sparse Mode, on the other hand, assumes that no client in the network is interested in being in the multicast group, and if it is, it needs to send a join request.

Communication between hosts and routers is implemented by another set of protocols, from which the Internet Group Management Protocol (IGMP) [9] [10] and Multicast Listener Discovery (MLD) [11] are the most used. These protocols enable clients and sources to inform the neighbor routers that they are interested in joining a specific multicast group. Upon the receiving of these requests, the routers will use the routing protocols mentioned above to configure the multicast tree, assuring the creation of paths between the source and the existing clients.

However, despite the fact that IP multicast is capable of a significant improved management of the network resources, ISPs are reluctant to deploy and provide multicast over the Internet [12]. The main reasons for this are related to the serious scalability issues over large-scale networks and the requirement of a global deployment of multicast-capable routers. Furthermore, the lack of access control support and the inexistence of an appropriate pricing model, also contributed to the stall of the commercial multicast implementation.

In order to solve these issues, more flexible solutions as the Application Layer Multicast (ALM) [13] [14] [15] were addressed. ALM systems implement multicasting functionality at the application layer, instead of the network routers, overcoming infrastructure deployment issues. It enables the transmission of a single copy-packet per flow between end hosts, independently of the underlying network between them, adding support for IP multicast where it does not exist in the network. However, performance issues are introduced, such as increased delivery delays.

2.2.1. Multicast Mobility Solutions

NGN will focus on providing a heterogeneous environment, enabling the widespread of mobile terminals, which require constant network connectivity. Thus, it is necessary the existence of an efficient mobile protocol, capable of assuring that the handover can be executed seamlessly and without downgrading the service provided.

However, native support for mobility is not implemented in the majority of the IP multicast protocols. Hence, the usage of certain techniques is necessary to provide a proper integration of IP multicast with mobility. Among the available solutions, the most used are the Bi-directional Tunnelling (BT) [16] and Remote Subscription (RS) [17], although mixed approaches, named Agent-based solutions, are also used [18].

The BT solution proposes a mechanism in which the home agent (HA) that resides in the home network of the mobile node (MN) acts as the multicast source for the multicast tree. To send a join request to the multicast group, the MN starts by creating a bi-directional tunnel to its HA, through which the join request is sent. After the HA receives the request, it forwards it to the local multicast router. Therefore, after the successful join to the requested multicast group, every packet sent or received by the MN is forwarded through the bi-directional tunnel created with the HA. Thus, this mechanism provides a complete transparency of the MN mobility, allowing the multicast tree to remain static after the MN handovers. However, due to the fact that the multicast packets have to always be forwarded through the bi-directional tunnel, a significant increase on the delay of the data delivery is introduced, as well as an increased overhead in the network, introducing a considerable loss of performance.

On the other hand, the RS solution proposes a completely different approach, in which the MN re-subscribes to the previous multicast tree every time it enters on a foreign network. Hence, this solution provides an optimal routing strategy, since the multicast packets are always sent directly from local multicast router to the MN. However, since the tree has to be reconfigured each time the MN network changes, a significant message overhead is introduced. Furthermore, each time the MN network changes, it has to re-initiate the multicast transmission subsequent to the handover, resulting in a rigorous service disruption.

Another approach, based on the usage of Agent-based solutions, attempts to balance the disadvantages of both solutions presented above, by mixing some of their mechanisms. In this approach, Multicast Agents (MA) join the multicast tree in several different networks on behalf of multicast listeners. Thus, when the multicast source moves to a foreign network where an MA is present, it is not necessary to re-establish the existing multicast tree, since the MA was previously instructed to join the respective multicast group. Hence, it is only necessary to create the new path between the IP multicast source and the MA. Therefore, this solution provides a significant reduced overhead on the network, due to the considerably smaller necessary modifications to the multicast tree. Furthermore, since the multicast packets are always sent directly from the IP multicast source, the delay on the data packets delivery is also smaller. However, these protocols cannot be directly implemented on IPv6 mobility scenarios, since Mobile IPv6 does not support foreign agents (FA).

2.2.2. Multicast Resource Allocation Mechanism

Despite of the superior resource management of the IP multicast protocol in comparison to unicast, the integration of QoS mechanisms in IP multicast is still necessary to assure the expected quality levels by the users on the multicast transmissions.

Resource allocation in IP multicast is done in a per-flow basis, and created on-demand, meaning that resources are allocated to each micro-flow every time a new join-request is received. However, the fact that the reservations are implemented per-flow can pose serious scalability issues in large-scaled networks, decreasing the forwarding performance in the network routers, as well as a significant increased control overhead.

Thus, to provide multicast communications with QoS assurance, solutions to control the allocation of multicast resources and QoS are necessary. However, these are generally implemented separately, due to contradicting deployment concepts. The integrated deployment of DiffServ [19] and IP multicast is promising, since the former allows a scalable QoS approach, while the latter saves bandwidth by preventing packet duplication. On the other hand, their integration is not trivial, since DiffServ achieves scalability by pushing the complexity to the edge routers, while IP multicast is exactly the opposite, operating on a per-flow basis throughout the network. On the other hand, the dynamic addition of new users to the multicast tree using DiffServ will cause a multicast tree re-arrangement, imposing a negative effect on the QoS levels of the other receivers, in case that resources are not explicitly allocated for the new user. This problem is called Neglected Reservation Sub-tree (NRS) [20] and is responsible for the degradation of the quality of communications that have correctly reserved their resources.

On the other hand, not only QoS requirements have to be considered while creating the multicast tree. Factors such as routing asymmetries also play a fundamental factor [21], and are usually ignored by the majority of IP multicast protocols, due to the fact that these protocols build the multicast trees from the receivers to the source, while the data travels in the reverse direction. Thus, data packets are forward through paths that are not optimized for that purpose, leading to a loss of performance and a consequent failure to deliver the quality levels requested by the users. This may be caused by an array of distinct possibilities, such as different paths for both directions, same path but different resources available for each direction, as well as quality of service or network access restrictions.

The limited functionalities supported by DiffServ that are built to provide QoS guarantees require the implementation of external resource allocation mechanisms, since resource reservation and admission control is required at least to deploy Per-hop Behaviour (PHB) and services based on the Expedited Forwarding (EF) PHB. However, the integration of DiffServ and resource allocation mechanisms introduces performance issues. On one hand, the scalability achieved in DiffServ, allowing per-class control, is adverse to the performance degradation taken by per-flow resource allocation mechanisms currently deployed in the Internet through Resource Reservation Protocol (RSVP) [22]. On the other hand, the addition of new mechanisms increases system complexity and endangers its performance, which is not desirable. DiffServ was originally conceived for wired networks, but its aggregation concept was also extended for wireless networks.

To overcome these issues, two main solutions are presented: Overlay for Source-Specific Multicast in Asymmetric Routing (OSMAR) [23], which was designed to consider QoS and network asymmetries; Multi-service Resource Allocation (MIRA) [24], which enables the control of Class of Service (CoS) resources, also considering the network asymmetries.

OSMAR approach was designed to be used as an overlay for source-specific multicast protocols (e.g. PIM-SSM [25]), enabling them to provide content distribution considering QoS while also considering network asymmetries. This is accomplished by changing the MRIB table values. This table will then be used by the IP multicast protocols to build the multicast tree considering the path from the source to the receivers, and thus enabling the creation of optimal data paths, solving the reverse path problem which the majority of the multicast protocols are subjected to.

MIRA, on the other hand, is a possible solution to control the CoS resources in multicast trees, taking into account the asymmetric route problems. It provides the desired QoS levels by the user for each flow by adapting the resources of the respective CoS. Furthermore, it supports the creation of QoS aware multicast trees by manipulating the MRIB. The update of the MRIB and CoS bandwidth is done in a single operation from the ingress to the egress router in the direction of the access-router of the user. In the ingress and interior routers, the configuration of the correspondent CoS in the outgoing interface is indicated by the unicast RIB table. Then, the MRIB is updated with the IP address of the previous visited router. As for the egress router, the CoS configuration is done by taking into account the Service Level Specifications (SLS) established with the neighbour network. After the CoS configuration and MRIB update, the PIM-SSM is triggered by MIRA. The agents placed in the interior routers only store essential information, as the per-class reservations, enabling an optimized control and packet forwarding. Edge routers, on the other hand, store a significant amount of information as a list if the involved interior routers for the reservation paths, information relative to available CoS, information of the edge to edge per-class reservations, loss tolerance, among others. Although it may seem, at first sight, that the edge routers are overloaded with information, this technique makes possible that the interior routers store just the essential information, allowing for a significant increased packet forwarding optimization.

2.2.3. Multicast Overlay Routing

As explained, the usage of the IP multicast protocol on the current networks is poorly supported. Furthermore, IP multicast suffers from several specifications problems, like the small IP address range for the multicast groups. On the other hand, unicast is oriented to end-to-end communications, having poor resource management on situations that require the same content to be sent to multiple clients simultaneously.

To provide a solution for the IP multicast problems, the multicast overlay routing was investigated. It suggests the support for multicast functionalities through the creation of a backbone overlay of intermediate proxies, creating multicast trees between them. Communication (unicast or multicast) between end hosts is possible through these proxies. This mechanism introduces a superior scalability in multicast trees management and group membership.

Multicast overlay, unlike ALM, supports more than one group or service in the same overlay node. This behavior makes the multicast overlay routing more suitable for environments and applications with various groups. Furthermore, it has a better performance than ALM in control overhead.

Overcast [26] is an ALM implemented as an overlay network. It aims for a superior management of the network, maximizing the available bandwidth between the source and all clients. Its architecture is based on a set of proxies organized in a distribution tree placed in the central source for a single multicast source. The establishment of the source specific tree is achieved through a distributed tree-building protocol. The maximization of the available bandwidth is done through the usage of a self-organizing algorithm. A new node starts by initially choosing the root as its parent, performing then a series of searches to decide its best placement on the tree. The maximization of the bandwidth comes, however, at the price of possible increases in the delay value, and thus it may not be suitable for some multicast applications.

2.3. Context-Aware Networks

Context-awareness can be described as the capability of network applications to be aware of certain information relative to the users, terminals, network and environment. This ability is essential for future network architectures in order to provide not only a significantly better resource management on the network, but also to improve the quality of the services delivered to clients, taking into account all the context information.

This improvement is possible through the inclusion of context information in the network decisions. Context can be defined as any type of information that can be used to define the state of an entity [27]. This entity can be any element in the network relevant to the necessary network decisions, such as clients, client's terminals, network situation and the surrounding environment.

There are several types of context information. For example, context information relative to users can be the client preferences, behaviour and profile, essential to the improvement of the quality levels of the services delivered to them. Furthermore, other parameters such as the client's location and trajectory can also be important to provide a more personalized service. Still on the client side, information about its terminal is also fundamental, since it can provide the supported codec list, available interface list, among others. Network information is also essential to provide the selection of the best path to deliver the requested contents to various clients. Thus, information such as available network resources, topology, link state and performance feedback are important, since they are relevant factors in the selection of the best available path. Environmental context information can also be used to provide a more personalized service to the clients. For instance, by grouping environmental context information such as user location and the location temperature, it is possible to know if the user is currently in an outdoor environment or indoors. Hence, the WI-FI scanning could be turned off/on accordingly, providing an improved battery management of the client's terminals.

It is then clear that the usage of such information can provide not only a better management of the available resources, hence improving the quality of service provided, but also to offer the same services in a more personalized manner, delivering the requested contents to the users considering not only the network conditions, but also the users preferences and capabilities, leading to better levels of quality of experience.

Having knowledge of the various context information available to be gathered is not enough to allow this information to be used by the network and/or applications. The first step towards the utilization of context information is the usage of elements such as sensors that are capable of collecting the available information. However, this collected data is not often in a suitable shape to be directly used. Thus, it is necessary to organize and format it in such a way that it becomes usable for other interested elements.

Lately [28] there has been research on the area of cognitive networks, capable of improving the performance of complex networks, with a high volume of dynamic context information available. A cognitive network is a network that is able to read the network conditions and adapt itself to them, learning from past modifications. Thus, it is capable of improving the end-to-end performance.

2.4. Signalling Protocols

In the last years, the Internet has assisted to a significant increase in the number of users, network size and supported applications (e.g. multimedia real time applications). This increase places an increasing demand for signalling protocols capable of enabling a seamless control communication between various applications. One example of this need is present in the Voice over IP (VoIP) service (Figure 3), where users behind firewalls or in networks protected by firewalls are unable to communicate with each other. Thus, the usage of a signalling protocol to install some firewall policies in the routers along the path is essential. Upon the successful installation of these policies, the hosts are able to successfully establish a connection.

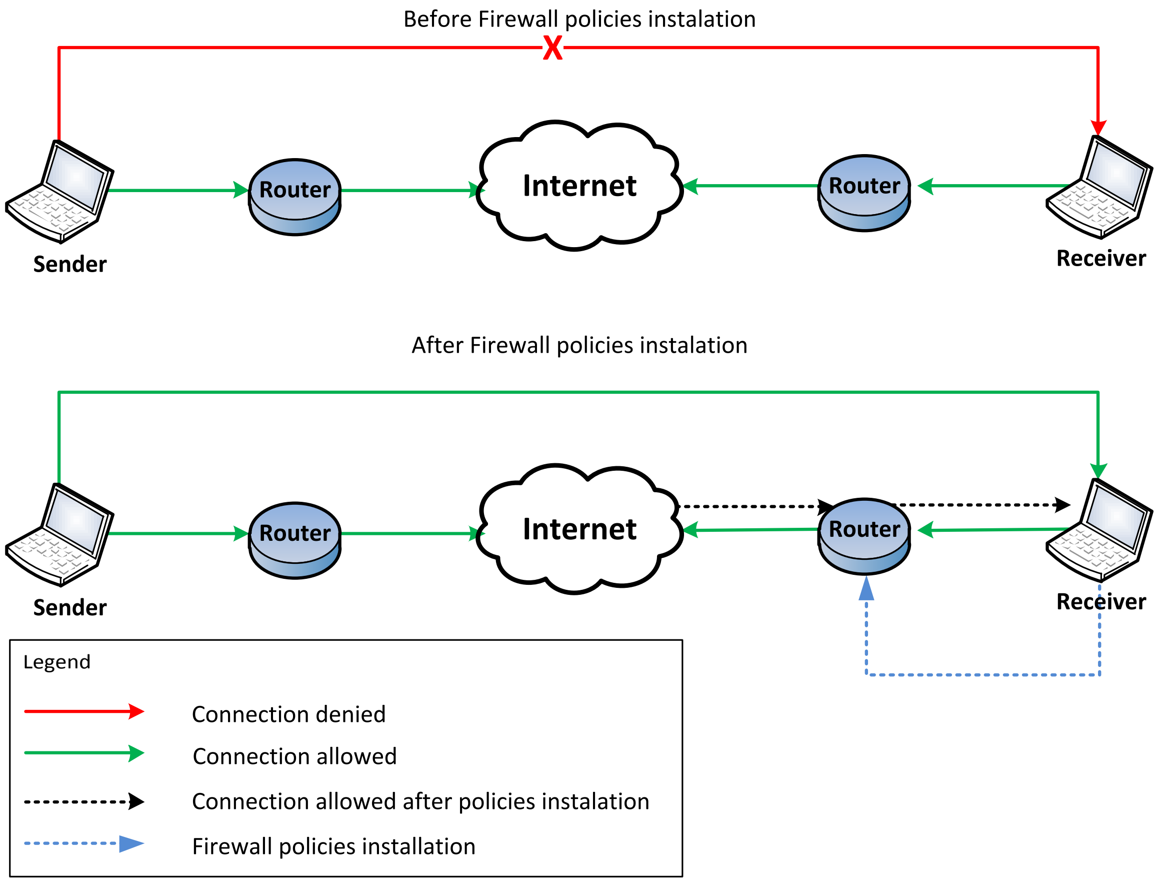

Before Firewall policies instalation
X
Internet
Router
Router
Sender
Receiver
After Firewall policies instalation
Internet
Router
Router
Sender
Receiver
Legend
Connection denied
Connection allowed
Connection allowed after policies instalation
Firewall policies installation

Figure 3 – Firewall signalling impact for VoIP

However, several other situations exist where signalling control between applications is necessary for a correct operation. Thus, the study of various signalling protocols is essential to the successful implementation of the proposed objectives of this dissertation.

2.4.1. Next Steps in Signalling Protocol (NSIS)

The Next Steps in Signalling (NSIS) [29] protocol is responsible for the standardization of an IP signalling protocol, primarily for QoS signalling, although support for signalling between various applications is also provided. This protocol consists of two layers (Figure 4), the upper layer NSIS Transport Layer Protocol (NTLP) and the lower layer NSIS Signalling Layer Protocol (NSLP).

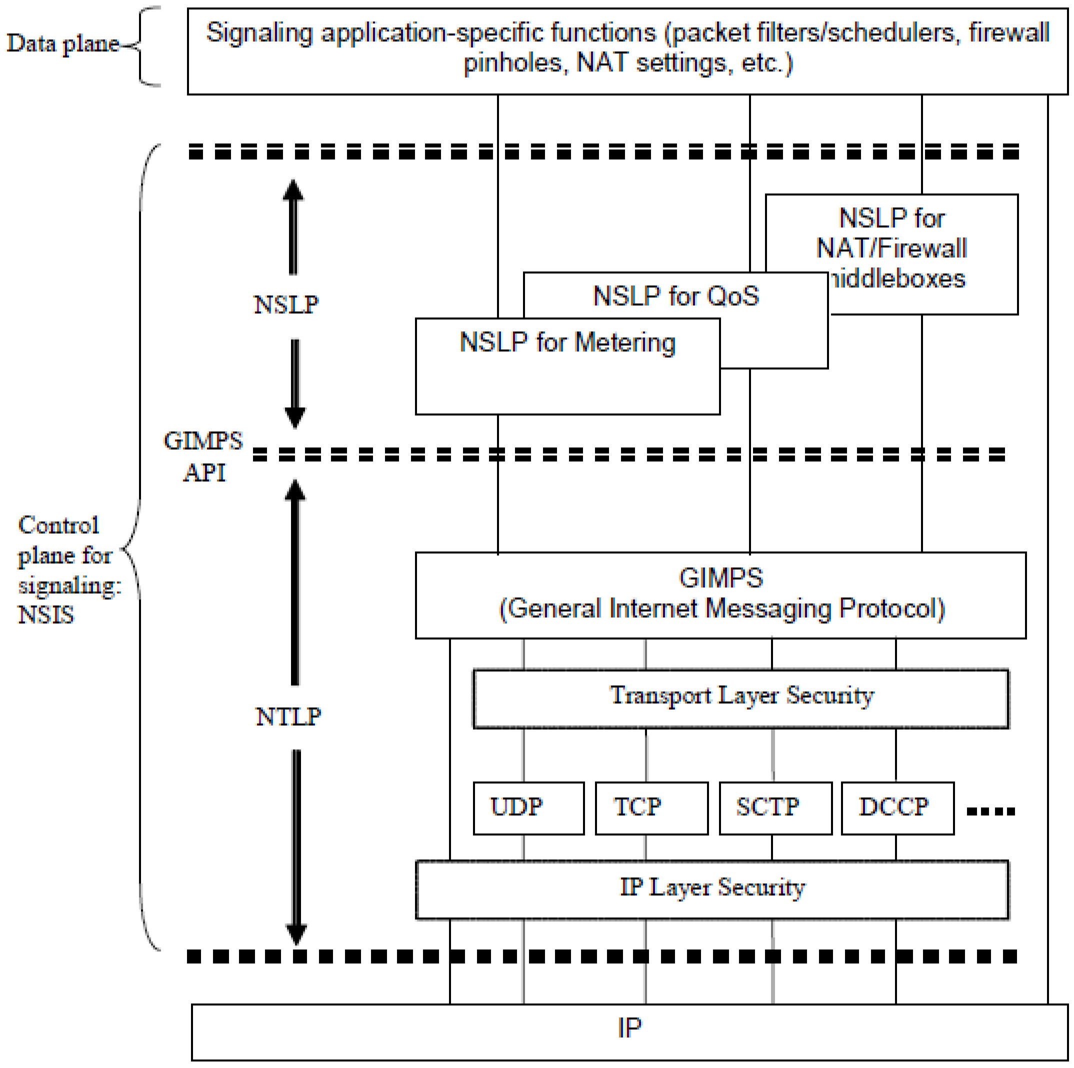

Data plane
Signaling application-specific functions (packet filters/schedulers, firewall pinholes, NAT settings, etc.)
NSLP
NSLP for NAT/Firewall middleboxes
NSLP for QoS
NSLP for Metering
GIMPS API
Control plane for signaling: NSIS
GIMPS (General Internet Messaging Protocol)
NTLP
Transport Layer Security
UDP
TCP
SCTP
DCCP
IP Layer Security
IP

Figure 4 – NSIS protocol stack [30]

The NSLP layer is designed for a particular signalling application, interacting on one side with the lower NTLP layer, and in the other with a specific signalling application. Furthermore, this layer may also define several rules for the messages format and/or sequence for a specific signalling application.

On the other hand, the lower layer NTLP is designed to interact with various NSLPs on the upper side, and with the IP layer on the lower side. Its main objective is to provide the transport of signalling messages sent by the NSLP between two NSIS nodes. However, it is also responsible for enabling the exchange of control information such as route modification and error messages. The NTLP is formed by two separate layers, the network transport layer such as UDP and TCP, and the core component General Internet Messaging Protocol for Signalling (GIMPS).

The GIMPS component is responsible to determine how to reach adjacent NSIS nodes, selecting the appropriate transport protocol and transferring the data. It is also responsible for deciding whether the received message from the underlying layer should be sent to the upper NSLP layer, or if it should be forwarded to the next NSIS node.

An NSIS entity (i.e. a network element that supports the NSIS protocol) may be responsible for one of the following roles:

- Initiator: the NSIS entity is responsible for the initialization of the NSIS signalling.
- Responder: the NSIS entity is the reception target of the signalling message, ending the NSIS signalling.
- Forwarder: in this situation, the NSIS entity is only responsible for the forwarding of the signalling message towards the responder.

Furthermore, the relation between various NSIS entities can be described as neighbours and/or adjacent peers. Every NSIS entity is said to be in a neighbour relationship with each other. However, to be in an adjacent relationship, the NSIS entities have to support the same signalling protocol (i.e. NSLP).

2.4.2. Common Open Policy Service Protocol (COPS)

The Common Open Policy Service Protocol (COPS) [31] is a standard that specifies a simple client/server model for supporting policy control over Quality of Service (QoS) signalling protocols. It is used between servers known as Policy Decision Points (PDP), where the policies are stored, and the clients known as Policy Enforcement Points (PEP), where the policies are enforced (Figure 5).

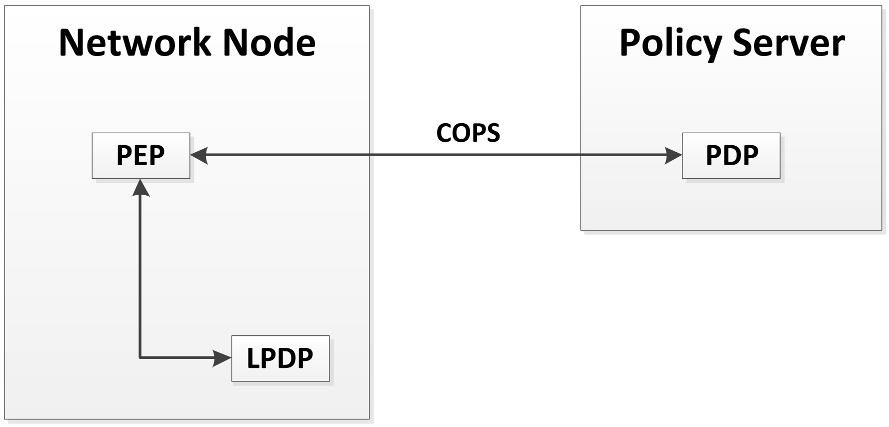


Figure 5 – COPS basic model

The COPS protocol specifies two different models. In the first model, named Outsourcing Model, all policies are stored at the PDP. Then, whenever the PEP needs to make a decision, it sends all relevant information to the PDP, where it is analysed. The decision based on the information received is then sent to the PEP, which is only required to enforce it. On the other hand, in the Provisioning Model, the PEP reports its decision-making capabilities to the PDP. The PDP then downloads relevant policies on to the PEP. The PEP is then able to make its own decisions based on the policies received. In this model, the Policy Information Base (PIB) is used as a policies repository.

2.4.3. Diameter

With the increase of the new services and applications, the requirements for authentication and authorization mechanisms have greatly increased. The Remote Authentication Dial-In User Service (RADIUS) [32] [33] protocol previously used can be insufficient for these new requirements. Thus, a new protocol that is capable of fulfilling new access control features while keeping the flexibility for further extension is essential.

The Diameter protocol, defined in the RFC 3588 [34], is an evolution from the mentioned RADIUS protocol, and it is generally considered to be “twice as the RADIUS”, since several and significant improvements were made in various different aspects. It is also generally believed to be the next generation Authentication, Authorization, and Accounting (AAA) protocol.

Diameter is implemented as a Peer-to-Peer architecture, meaning that every host where the Diameter protocol is deployed is able to act as a client or as a server. Thus, is it considered that the Diameter node that receives the connection request will act as the Diameter client.

The communication mechanism between the two Diameter nodes can differ between request types. However, in the general situation, the Diameter node will send a request message to another Diameter node acting as a server. The Diameter server proceeds to process the received request, and decides the appropriate actions to it. Then, if the request is successful, the Diameter server sends a response message to the Diameter client informing of the request success. On the other hand, if the Diameter server is unable to perform the actions requested by the client, an error message is sent instead. The described architecture might seem comparable to standard client-server architecture; however, it is also possible in some situations that the Diameter server is capable of acting as a Diameter client.

Nodes where the Diameter protocol has been deployed can be more than just clients or servers. These are named Diameter agents, and typically there are three kinds of Diameter agents, Relay Agent, Proxy Agent and Redirect Agent.

Relay Agent

The Diameter Relay Agent is used to route Diameter messages to the appropriate destination based on the information contained. This routing decision is performed using a list of supported realms and known peers. It is especially valuable since it can aggregate requests from different realms to a specific realm, eliminating the heavy configurations necessary of network access servers for every Diameter server change.

Proxy Agent

The Diameter Proxy Agent can also be used to route messages; however, unlike the Relay Agent, the Proxy Agent can modify the message content. Thus, they are able to modify these messages to implement policy decisions, such as controlling resource usage, providing admission control and provisioning. The usage mechanism of this agent is shown in Figure 6, where it is used to forward a message to a different domain. However, if the message content is not to be modified in the Diameter proxy agent, a relay agent would suffice.

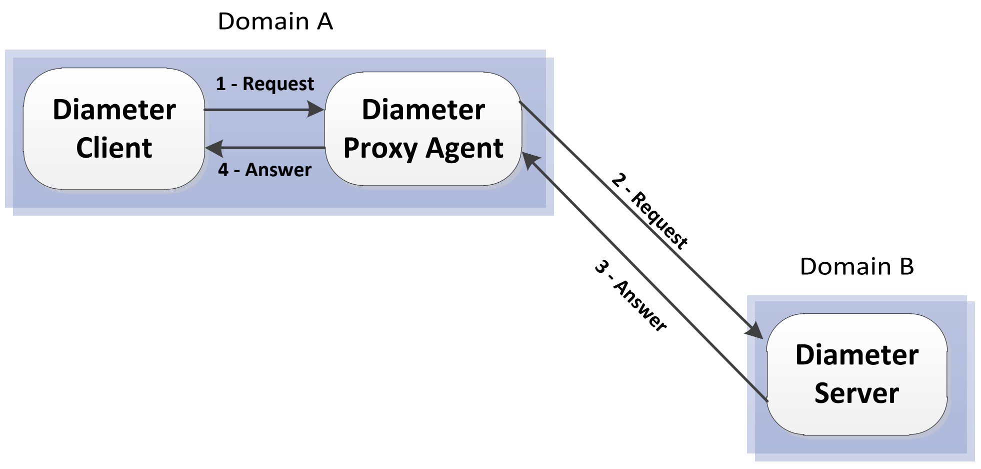


Figure 6 – Diameter Proxy Agent

Redirect Agent

Unlike other Diameter agents, the redirect agent does not forward the request messages. Instead, when the redirect agent receives a request message, it checks its internal routing table, returning a response message containing the redirect information to the peer that sent the original message. Thus, the use of this agent enables the other diameter nodes to not keep a local list of the routing entries. An example of the usage of a redirect agent is presented in Figure 7, where it is possible to observe that the redirect agent is out of the request message forwarding path. Since at this time the proxy agent is not aware of the address of the Diameter server, it is necessary to perform a request to the redirect agent to get the address.

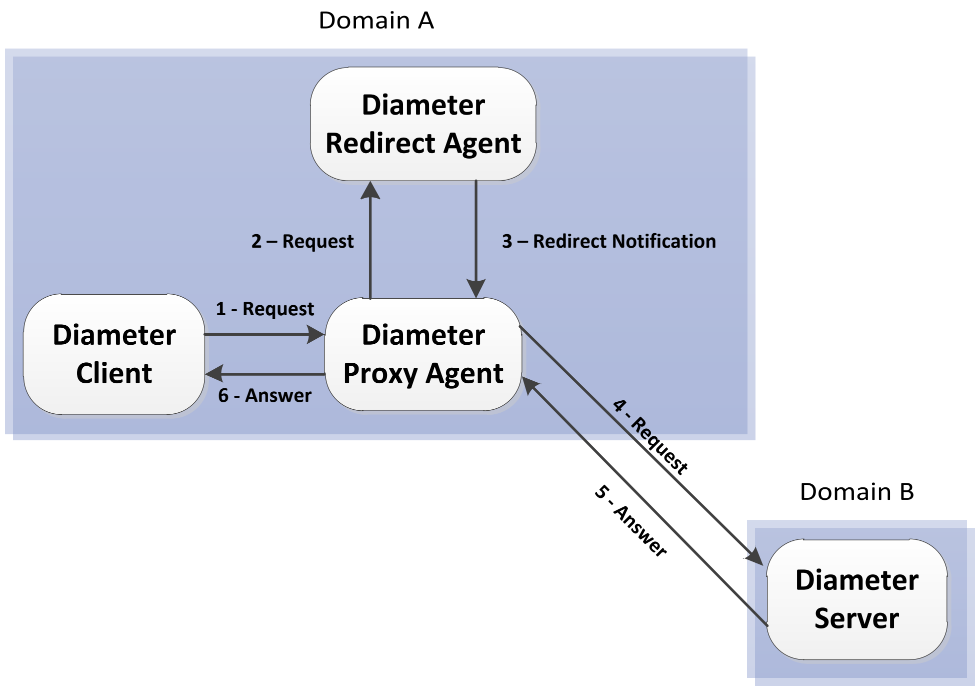


Figure 7 – Diameter Redirect Agent

The communication between various Diameter nodes is achieved through the exchange of data packets named Diameter messages. These messages are completely defined, as shown in Figure 8, and each message contains a value field to specify the request type. This value is then used by the receiver to identify what type of information is placed within the Diameter message. For each existing request type, there is a matching answer type sharing the same command ID.

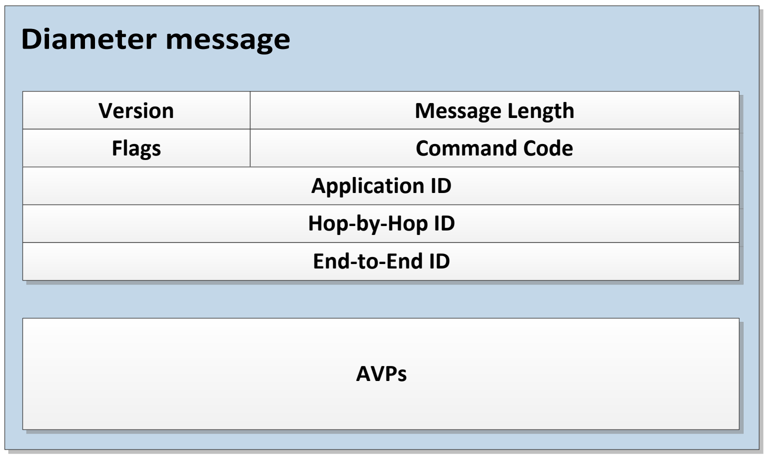


Figure 8 – Diameter message layout

The command information is placed within one or more Attribute-value Pairs (AVPs) (Figure 9). Several AVP Codes are defined by the Diameter protocol, and these should be used by new applications whenever possible. The data values placed in each AVP have to follow specific data types specified within the protocol.

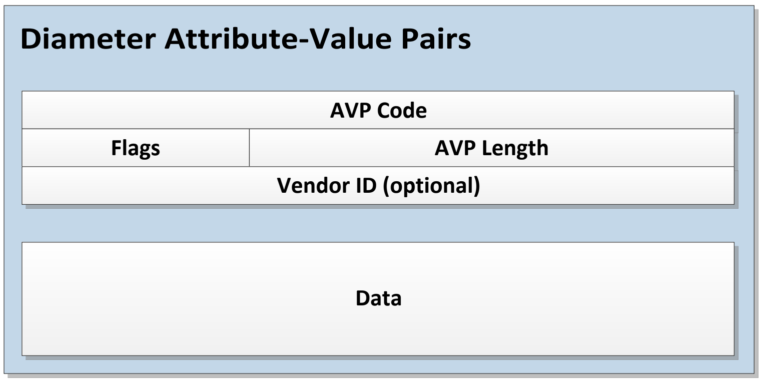


Figure 9 – Diameter Attribute-Value Pairs layout

2.5. Summary

This chapter presented an overview of some of the technologies and mechanisms used for the support of the C-CAST project and for the work developed in this dissertation.

A special study was done to the IP multicast protocol, which is essential for the work developed in this dissertation, where it is possible to confirm that the use of IP multicast effectively reduces the network load, granting an improved management of the network's resources. However, it was also verified that the implementation of the IP multicast on current networks is a challenge, since some essential features are not clearly supported, such as mobility and support for quality of service.

Context-aware networks were also studied, were it was analysed the impact that the usage of context information can have on the network performance. It was also identified what are the main sources of context information relevant to the network, the process this information has to be subjected to until it is ready to be used, and how it can be enforced in the next generation networks through the usage of cognitive networks.

Finally, various signalling protocols were also studied in order to provide the knowledge of how the exchange of control information in the developed component should be processed.

The combination of all the concepts studied here can be merged to create a new concept of multiparty groups of users with similar context information, to which various multimedia contents are going to be delivered in dynamic multimedia sessions. The study of the IP multicast protocol as well as multicast overlay routing provide mechanisms to improve the management of the network resources used in the delivery of the multimedia contents to the clients. The inclusion of relevant context information in the network decision process also enables the development of an architecture network capable of adapting itself to match the network conditions, and thus being able to deliver services with an improved quality of experience for the connected users, and also with assurance of the quality levels requested.

3. C-Cast

3.1. Organization

This chapter describes the general concepts and objectives of the C-Cast architecture, presenting the main components, their functionalities and interaction.

Section 3.2 describes the relevance of context-aware networks, as well as the general concepts and ideas behind the C-Cast architecture.

Section 3.3 presents a description of the context framework within the overall architecture.

In Section 3.4, a more detailed overview of the multiparty session management framework is presented, focusing in the necessary interactions between several components of the network to establish the user groups and support the specified session events. In this chapter, it is also explained how the network manages different capabilities/preferences among users in the same multiparty groups.

Section 3.4.1 specifies the multiparty transport framework, describing the necessary interactions and procedures to assure the correct enforcement of the multiparty trees on the network. It is also explained how the proposed architecture can achieve an independence of the underlying network to deliver the multimedia contents to the users.

Finally, section 3.6 presents examples of the required communications between the several network components for the main session events proposed.

3.2. General Concepts

The C-Cast project main objective is the development of an intelligent network, capable of retrieving context information from users and from the underlying network, using it to deliver personalized media content to users grouped together by similar context. Since the same content is being forward to multiple users at the same time, the multicast capabilities of the network should be used whenever possible, granting a better usage of the network's resources. However, the underlying network through which the content is being forward may not completely support multicast transmission (if at all) and as such, the developed network should be able to deliver the media content to interested users independently of the network capabilities.

Context awareness refers to the network's capability to gather information from the surrounding environment and adapt itself to it, providing a better and more personalized service delivery. The users are then grouped together by similar context information like user's interests, current location and similar social context. The media content delivered to these users will then be the same. However, smaller aggregations of users inside these groups are possible, since context information like terminal capabilities and the access network they are connected to might be different. So, despite the media content being delivered to all the smaller groups is the same, the QoS required by each one is different.

This information however, does not remain static through the complete session. Modifications to the user's context, for example, may require that the existing groups are re-organized to match the new current context values. This means that the network has to be capable of making intelligent decisions to adapt itself to possible modifications, even if these occur during the streaming of the media contents requests by the groups of users.

By grouping together all these characteristics, the developed network will be able to deliver media content to users based on their preferences and also adapt it to the current network state or even the user's terminals capabilities. Furthermore, the ability to update the user's context during a session guaranties that the media contents are being delivered to users according to their requirements.

On the other hand, the network's ability to retrieve context information and consecutive capability of grouping together users with the same context allows that several streams can be aggregated in multicast streams, granting a significant increase in the system scalability.

3.3. Context Detection and Distribution Framework

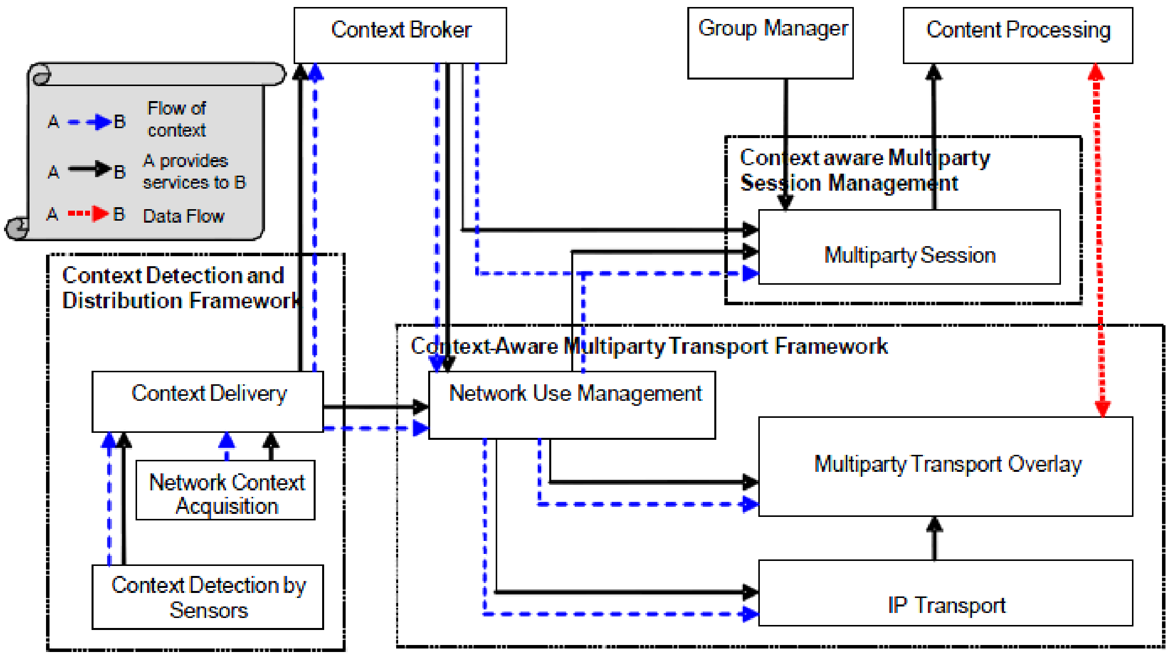

Context Broker
Group Manager
Content Processing
A → B Flow of context
A → B A provides services to B
A → B Data Flow
Context aware Multiparty Session Management
Multiparty Session
Context Detection and Distribution Framework
Context-Aware Multiparty Transport Framework
Context Delivery
Network Use Management
Multiparty Transport Overlay
Network Context Acquisition
Context Detection by Sensors
IP Transport

Figure 10 – Context Detection and Distribution framework layout [35]

One of the main objectives of the Context Detection and Delivery framework (Figure 10) is the detection and collection of available context information in the network and environment through the use of sensor components called Context Providers (CxP). These components are also responsible for detecting the user context information, like preferences, supported codecs by the user's terminal, information of the access network the user is connected to, among others.

The collected information is then used by several components on the network to provide a considerably more optimized set of decisions, enabling an improved usage of the network's resources and the assurance of delivering the quality levels expected by the users. On the other hand, this information can also be used to grant a more personalized service, by delivering the preferred contents in the preferred/supported formats to the users, hence increasing the QoE provided. The elements in the network capable of using the context information gathered are designated by Context Consumers (CxC).

The information gathered by the various CxPs is not directly delivered to the CxCs. Since the CxPs are scattered through the network, the information is not accessible in the same place, forcing the CxCs to establish several connections to retrieve the information from several sensors simultaneously. Furthermore, the CxCs would be obliged to know in which CxP the desired information was stored. Thus, to address these issues, the Context Broker (CxB) was designed. This unit, together with the Context History (CxH) provides a centralized storage for the entire context information gathered by the CxPs. The retrieval of the context information by the CxCs is then greatly simplified, since only one communication has to be established to gain access to the complete context information of the network, environment and users.

3.4. Context-Aware Multiparty Session Management

The groups of users are created by the Group Management Enabler (GME) module, which is responsible by evaluating the information provided by the context broker (and in some cases by the CxP directly), joining different users with similar capabilities and preferences in the same group. Users can also be joined in groups by approximate context parameters like preferences, social network, terminal's capabilities, etc. The multimedia contents delivered to the user groups are then selected by the Content Selection Enabler (CSE) based on the group's content preferences and in the rules of the application that invokes the CSE.

However, despite the fact that the same multimedia content is going to be delivered to every user in the same group, not every user is going to be able to receive that content in the same format and/or quality. This is due to the fact that the terminals used by the users are most likely different, and as such, they possess different capabilities and requirements. This issue is addressed by the Session Management Enabler (SME) module, which is responsible for determining the supported formats for the media content selected for each user in the same group. The list of selected formats is then forwarded to the Content Processing and Delivery (CtPD).

The selection of the preferred/supported formats by the users is not sufficient to guarantee the requested QoS. The network situation has also to be considered. Thus, if the network is overloaded, it is preferable to deliver a certain format with lower quality and lower bitrate, respecting the delivery guaranties expected by the user, than delivering high quality content, but with poor or no delivery guaranties at all. This selection is then accomplished by the Network Management Enabler (NME), which is responsible for addressing the network context information and selecting the more appropriate bit rate for every user in each group. After the more appropriate bitrate for the content is selected, it is essential to make the necessary reservations in the network to ensure the delivery of the multimedia content with the expected quality levels.

In the multiparty framework, it is also important to mention the Session Use Management (SUM). This component is responsible for the complete session SIP signalling that enables the multimedia contents to be delivered to users. It supports several event types, namely Session Creation, Session Modification and Session Removal. The complete explanation of the message exchange for these events is going to be presented in section 3.6.

3.4.1. Context-Aware Multiparty Transport Framework

The context-aware multiparty transport framework (Figure 11) is composed by the Network Use Management (NUM), the Multiparty Transport Overlay (MTO) and by the IP Transport (IPT). Their main responsibility is to select the paths in the network through which the contents should be streamed and enforce them.

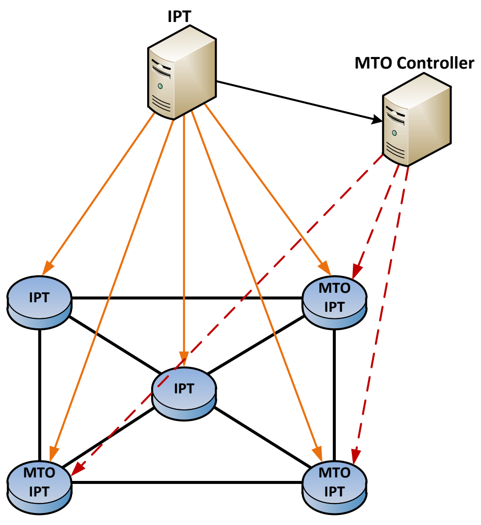


Figure 11 – Context-Aware Multiparty Transport framework layout

The NUM component is responsible for the selection of the best paths available on the network to deliver the multimedia contents to the user's terminals. It achieves its goal through an evaluation of the network conditions, based on the network context information, allowing it to select the paths between the streamer source and the terminal that guarantee the QoS levels requested. After selecting the complete network setup for the session, it sends this data to the IPT in order to successfully enforce it on the network.

The Multiparty Transport Overlay (MTO) is a generic transport service for group communications. This service enables an independency of the underlying networks in terms of IP multicast capabilities and IPv4/v6 support; thus allowing any user to participate in a multiparty delivery session independently of the network he/she is attached to and in a transparent manner to the application. The MTO fragments the concept of Abstract Multiparty Transport (AMT) in order to introduce the concept of sub-AMTs (Figure 12), which can be seen as sub-networks formed between each pair of Overlay Nodes (ON). This concept allows for an increase in the network's scalability and reliability, enabling the MTO to treat each Sub-AMT separately from the others. Thus, if an AMT lacks the support for multicast IP between two specific ONs, the MTO sends the data packets through unicast connections in that Sub-AMT, while for the remaining Sub-AMTs the data packets are sent by IP multicast, and the establishment of a path between the two ONs in each Sub-AMT is responsibility of the multicast routing protocol used.

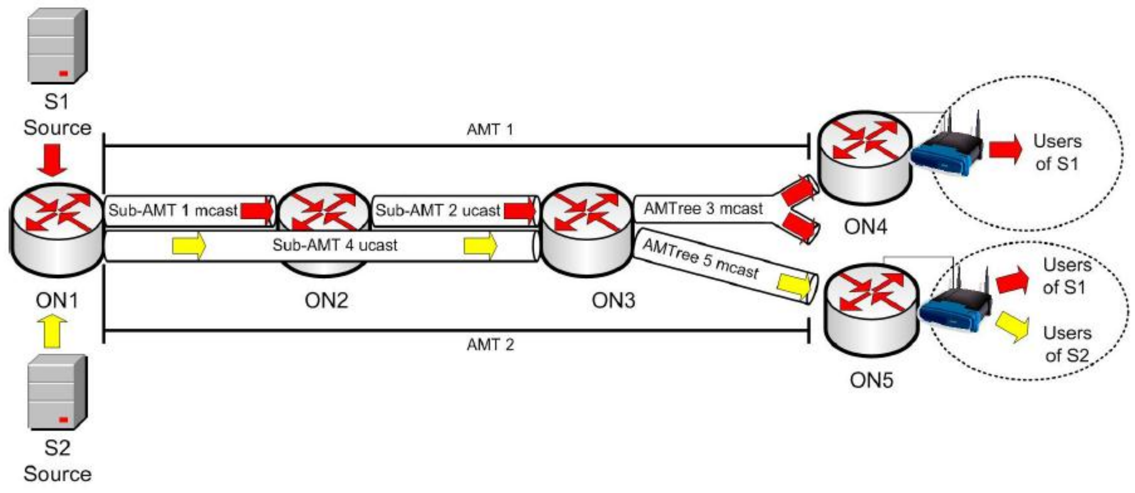


Figure 12 – AMTs and Sub-AMTs [36]

The IP Transport (IPT) is the core network component responsible for the enforcement of the multicast paths on the network. More specifically, it is responsible for the creation of multicast paths in the Sub-AMTs created by the MTO where IP multicast is supported. This component loses the abstraction between unicast and multicast paths, since it differentiates the connection types that have to be enforced. It acts as an interface between NUM and MTO, processing NUM requests and commanding MTO to enable the successful enforcement of the unicast paths and overlay functions. It is also responsible for assigning IP addresses of the created multicast groups, maintaining a list of the available and used IP multicast addresses.

3.5. IP Transport

The IPT is responsible for maintaining a direct interface with the NUM component, through which the various requests to enforce the selected trees in the network are sent. These requests are then internally analysed, and then enforced in the network through the IPT Node or in the MTO. A smaller part of the global IPT component, namely the IPT Node module, is responsible for the complete management of the IP multicast transport, through the creation, maintenance and removal of the various existing multicast groups in the network.

The IPT component is responsible for the complete management of the multicast traffic, providing a reliable interface with the MTO through which it is possible to control the underlying overlay network.

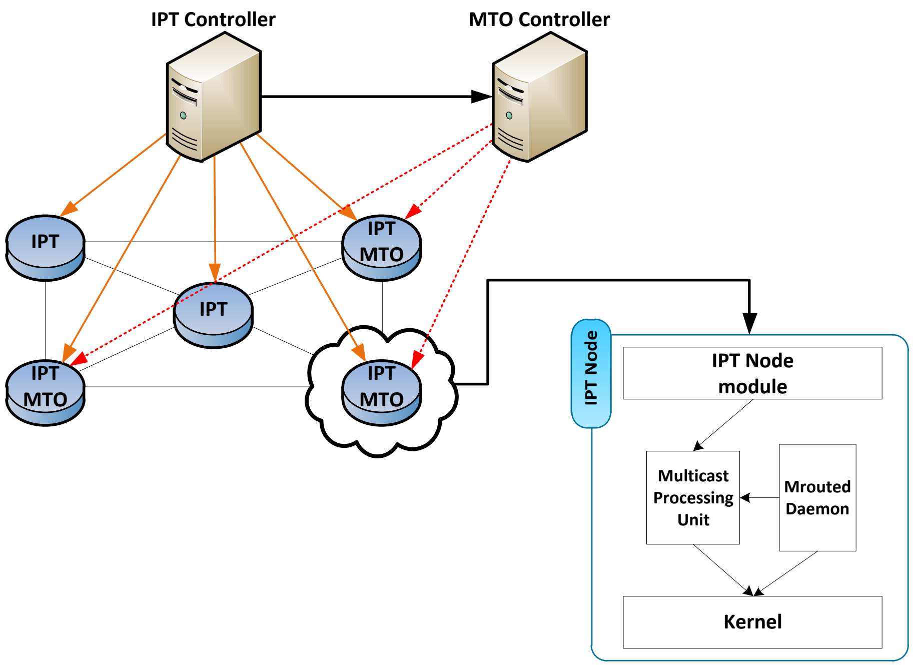


Figure 13 – IPT Architecture

The IPT is implemented in a hierarchal structure, in which a centralized unit (IPT Controller) is responsible for the complete set of necessary intelligent decisions, while another module (IPT Node) deployed in several nodes along the network is responsible for the enforcement and management of the multicast transport.

3.5.1. IPT Controller

The IPT Controller module provides a very important interface for NUM. It is only through this module that the NUM is able to enforce the trees selected for the various streams.

The interface with the NUM component provides the support for several session events, namely session setup, session removal and session modification. It also supports a smaller event for updating the port numbers through which the unicast terminals are receiving the multimedia content stream. Upon receiving a new request from NUM, it is provided to the IPT Controller a set of important data grouped in a single class object. It is in this class that the IPT Controller finds necessary tree information such as the nodes through which the stream is going to be sent, interested users, the connection types (i.e.

unicast or multicast), etc. The IPT Controller is then responsible for interpreting this data and converting it to several commands to the MTO Controller and the various IPT Node modules.

It is important to notice that the enforcement of these requests on the network is not performed by a single component, but instead, it is performed simultaneously by the MTO and the various IPT Node modules on the network. While the MTO is responsible for the enforcement of the overlay nodes with unicast connections, the IPT Nodes are responsible for the management of the multicast transport between the overlay nodes. Thus, nodes with only multicast connections are completely managed by the IPT Node module, while nodes with at least one unicast connection are managed by the MTO component. It is also important to notice that the IPT, namely the IPT Controller central unit, is responsible for the entire management of the multicast groups on the network, as well as for the dynamic allocation of IP multicast addresses for those groups.

The interface with the MTO Controller is based on the exchange of control messages through a socket connection. It is in these messages that the IPT Controller specifies the actions to be taken by the MTO relatively to the path tree nodes, connections and users. These messages have a very strict format, meaning that specific data has to be placed in specific slots, even though the message has a dynamic size. The details on the creation of these messages are explained in section 4.3.2. This interface is, however, bi-directional, meaning that the IPT Controller is required to wait for the MTO response, informing of the status of the requested actions (if the MTO Controller was able or not to accomplish the requested actions) and providing the IPT Controller with some essential data such as terminal's port number values to which the multimedia content packets should be sent by the leaf overlay node.

The interface with the various IPT Node modules requires the support to manage simultaneously several units throughout the network. Thus, it is obvious that a solution has to be found to provide the IPT Controller with complete knowledge of every active IPT Node module in the network. Hence, a registration mechanism in the IPT Nodes was created, in which, at the IPT Node module start-up, a message is sent to the IPT Controller with the purpose of adding a new entry of this node in the IPT Controller database. Thus, the IPT Controller contains a complete knowledge of every IPT Node module active in the network, and some information relative to that node, such as the node's interfaces, and which IP address can be used to establish a connection with it. Since new register messages can arrive at any time, it is essential that the IPT Controller is always able to receive and process them. Thus, it was necessary to create a parallel thread to this module, responsible for the management of the IPT Nodes information. Hence, if during an internal processing of a NUM's request the IPT Controller needs to send specific command instructions to be taken by the IPT Nodes, that information is sent to this separate thread. This thread is then responsible for grouping this data into a single Diameter protocol message and sending it to the known address of the desired IPT Node.

As mentioned in the beginning of this section, the IPT Controller supports various event types. The creation and removal of a tree for a specific stream provides the mechanism to start a completely new tree to deliver the multimedia content to specific terminals, while the session removal allows for a complete removal of an existing enforced tree. However, the IPT Controller also supports modifications to existing sessions, providing a mechanism for the network to react to certain changes (e.g. link down/up, user left/joined the multimedia stream) without causing a complete service disruption to existing users, by removing the existing tree and creating the new modified one. Hence, the support for this event allows to maintain a completely functional service to users who were not modified (either directly or indirectly), while at the same time, enables an entire adaptation of the network to several types of modifications. This mechanism is essential to the C-CAST architecture, since it is predicted that the various context information collected can change during the existence of a session, and as such, the network must be able to adapt to those modifications to maintain the quality levels expected by the clients.

3.5.2. IPT Node

Unlike the IPT Controller module, the IPT Node is deployed in several nodes throughout the network, providing a distributed mechanism for creating, maintaining and removing the multicast groups assigned by the IPT Controller, in order to provide a reliable multicast transport through specific paths.

Once the IPT Controller has decided which multicast groups are going to be created, and to which paths they are going to be assigned, it sends the necessary control messages to every targeted IPT Node module through a previously established socket connection. Each message sent contains the complete information of which is the targeted node's interface and the multicast group to which it is going to be associated. Since the IPT Node module is responsible for both creation and removal of multicast groups, it is also necessary to specify the message type, more specifically, if it is a join-request or a leave-request.

Since the communication between these two modules is done through the signalling protocol Diameter, each data value sent has to be included in a separate Attribute-value Pair (AVP) within the same message. Thus, for the mentioned communication with the IPT Node module, it is necessary to create three different AVP fields, one for each of the necessary information fields mentioned above.

After the message has been sent through the socket connection, the IPT Node module de-encapsulates every AVP included in the diameter message received. The data values gathered from each AVP in the diameter message are then used to enforce the requested multicast groups on the targeted node's interface. The multicast enforcement on the network is done through the creation of a new socket, immediately modifying the socket parameters to match the information received in the Diameter message. The maintenance of the multicast groups is then achieved through the *mrouted* daemon [37], capable of maintaining the multicast routing tables.

Thus, this allows that specific interfaces along several nodes on the network are added to the multicast groups requested by the IPT Controller, which will create the multicast tree specified by the NUM component. Hence, the multicast content data packets are forced to travel along specific paths defined by the various IPT Node modules.

3.6. Session Events

As mentioned before in section 3.4, the developed architecture provides support for various session events, such as Session Creation, Session Modification and Session Removal. It is through these events that the network is able to deliver the requested multimedia contents to the users. Thus, it is essential to have a complete understanding of their behaviour.

3.6.1. Session Creation

The session creation event (Figure 14) is used to create a new session of multimedia content delivery to a group of users. Relatively to the multiparty transport framework, this event is associated with the creation of a completely new tree.

The first step towards the creation of a multimedia session is the selection of the contents to be streamed to the user group. This selection is performed by the Content Selection Enabler (CSE). After the list has been selected, it is sent to the SME. After receiving the content list to be streamed, the SME is responsible to select the more appropriate codecs and formats for each user. This is done through a match between the context information of the group (provided by the CxB) and the available list of formats and codecs (provided by the CtPD) for the content list provided. This match provides the SME with mechanisms to adapt the contents that are going to be stream to each user to what they support/prefer, and thus providing a more personalized service. It is also able to adapt the content delivered to the users accordingly to their context information. One example of this situation can be given with a user in a noisy environment. In this situation, streaming both video and audio to the user is a mistake, since the user will not be able to hear it correctly. Thus, the SME will replace the audio stream by subtitles.

The list with the codecs and formats selected by the SME is then sent to the NME. Here, another selection will take place, but this time in a more network oriented point of view. Unlike the SME, where the files were selected based on the user context information, the NME performs a selection based on the network's context information. Thus, upon receiving the list from the SME, the NME will request to the CxB information on the network situation, such as bandwidth used, delay, etc. Then, combining that information it selects the more appropriate bitrate for each of the files selected by the SME. Hence, the selections executed by the SME and NME provide mechanisms to adapt the contents being streamed to the capabilities/preferences of the user terminals, and also, they provide mechanisms to adapt the quality of the files transmitted considering the network situation. Through these procedures, it is possible to improve the quality of experience provided, and at the same time, assure the quality of service levels requested.

After these selections, it is necessary to select the path through which the contents should be delivered. This task is achieved by the NUM component, which will evaluate the network and select the best available paths between the streamer and each user in the group. Upon the selection of the tree to be enforced in the network, the NUM sends this information to the IPT.

Upon successfully receiving the tree information selected by the NUM, the IPT (more exactly the IPT Controller) will send several commands to the various IPT Node modules deployed through the network and/or to the MTO. Since the IPT Node module is only responsible for enabling IP multicast transport, only nodes without unicast connections are enforced by the IPT Node. Thus, nodes with at least one unicast connection are not enforced by the IPT Node module, but instead, should be enforced by the MTO component. After the necessary actions are sent to both the MTO and the various IPT node, the network enforcement is complete and the MTO will return in an acknowledge message the port numbers that should be used, except for the connections to unicast terminals. A successful return message is then sent from the IPT to the NUM and then to the SME.

After receiving the confirmation of the successful enforcement of the tree on the network, the SME will send several SIP invites to the user's terminals. The unicast terminals will then return the port number to which its content should be delivered, while the multicast terminals will join the multicast group assigned by the IPT. At this moment, the content delivery to the multicast terminals is ready to start. However, for the unicast terminals, it is still necessary to inform the Leaf Overlay Node (LON) to which port number the multimedia contents should be sent. Thus, in this situation, the SME will perform two simultaneous actions. First, it will send a request to the CtPD in order to start the content streaming to the multicast users, and at the same time, it will trigger another session event type to update the port value of the unicast terminals. This trigger is received by the SME and forwarded to the IPT. The IPT will then send the necessary actions to the MTO to update the port number of every unicast terminals in the group.

After this last event is complete, the streamer has already begun the content transmission and now both terminal types (unicast and multicast) are able to receive the multimedia contents.

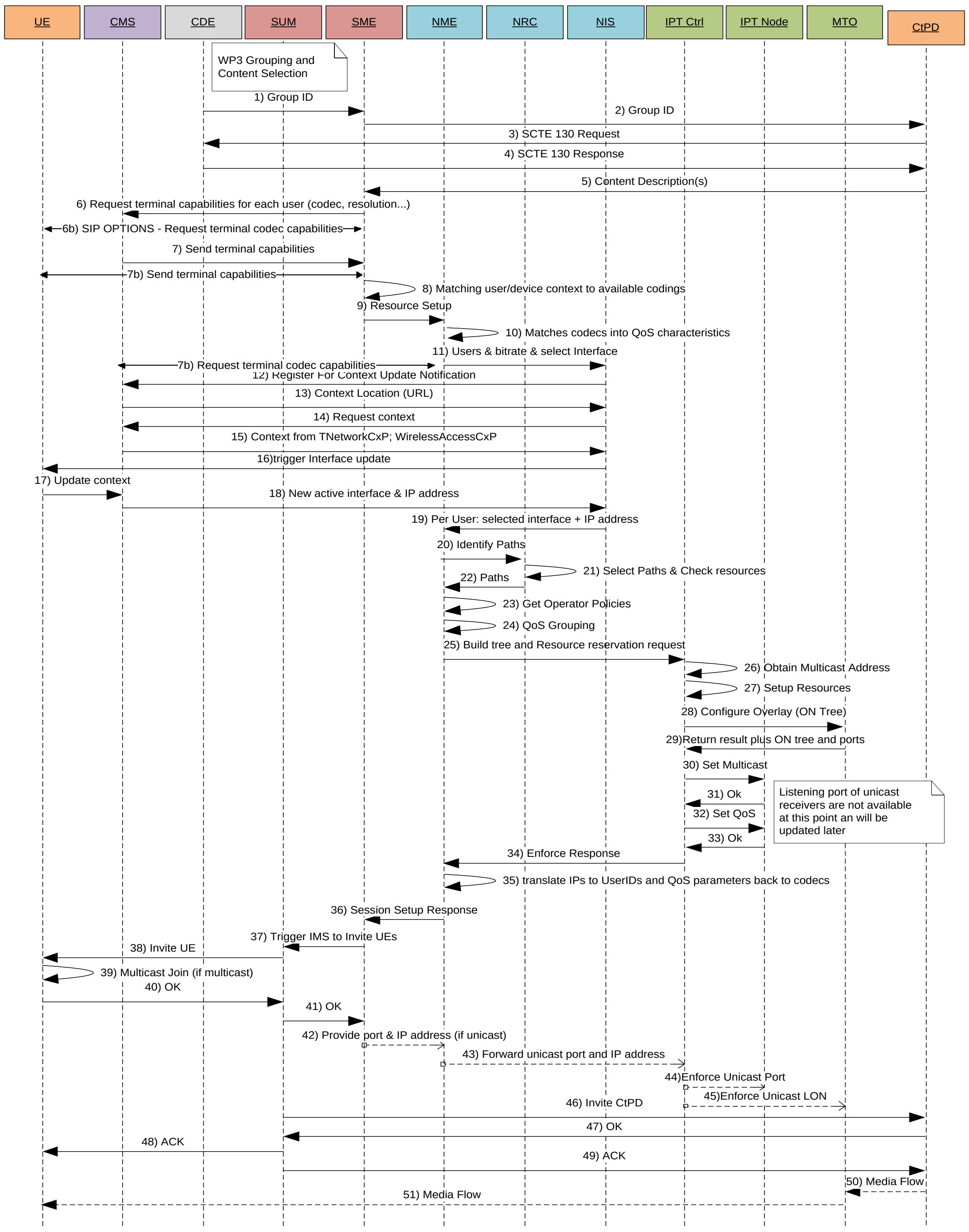

UE
CMS
CDE
SUM
SME
NME
NRC
NIS
IPT Ctrl
IPT Node
MTO
CtPD
WP3 Grouping and Content Selection
1) Group ID
2) Group ID
3) SCTE 130 Request
4) SCTE 130 Response
5) Content Description(s)
6) Request terminal capabilities for each user (codec, resolution...)
6b) SIP OPTIONS - Request terminal codec capabilities
7) Send terminal capabilities
7b) Send terminal capabilities
8) Matching user/device context to available codings
9) Resource Setup
10) Matches codecs into QoS characteristics
11) Users & bitrate & select Interface
7b) Request terminal codec capabilities
12) Register For Context Update Notification
13) Context Location (URL)
14) Request context
15) Context from TNetworkCxP; WirelessAccessCxP
16)trigger Interface update
17) Update context
18) New active interface & IP address
19) Per User: selected interface + IP address
20) Identify Paths
21) Select Paths & Check resources
22) Paths
23) Get Operator Policies
24) QoS Grouping
25) Build tree and Resource reservation request
26) Obtain Multicast Address
27) Setup Resources
28) Configure Overlay (ON Tree)
29)Return result plus ON tree and ports
30) Set Multicast
31) Ok
32) Set QoS
33) Ok
Listening port of unicast receivers are not available at this point an will be updated later
34) Enforce Response
35) translate IPs to UserIDs and QoS parameters back to codecs
36) Session Setup Response
37) Trigger IMS to Invite UEs
38) Invite UE
39) Multicast Join (if multicast)
40) OK
41) OK
42) Provide port & IP address (if unicast)
43) Forward unicast port and IP address
44)Enforce Unicast Port
45)Enforce Unicast LON
46) Invite CtPD
47) OK
48) ACK
49) ACK
50) Media Flow
51) Media Flow

Figure 14 – Session Initiation message sequence [38]

3.6.2. Session Modification

One of the requirements of the C-Cast architecture is that existing sessions can be updated in order to adapt to context modifications related to network/users/environment. Users that are receiving media content can be subjected to changes in the network they are connected to (device handover, network condition changes, etc) forcing the content being delivered to be updated to match these modifications. On the other hand, during a session, new users interested in receiving the same media contents can appear, while existing users might wish to stop receiving them. Network modifications can also occur, in the form of a link down/up, for example. Thus, it is necessary that the specified architecture is able to support the modifications experienced by the existing sessions.

3.6.2.1. New User

The addition of a new user to an existing session requires a new set of communications between the various components in the network to update the existing tree and include the new user in the delivery path of the multimedia contents.

The update starts by the SME being informed that a new user is added to an already existing session. Since the content list that should be sent for the existing group is already known by the SME, it is only required to have knowledge of the user's terminal capabilities/preferences in order to determine the supported codecs for the new user. This information is requested to the same component as before, the CxB. The list of the new supported codecs/formats is then sent to the NME, which will re-evaluate the network conditions based on the network context information and select the content files with the more appropriate bitrate value. The NUM is then triggered to select the new path between the streamer and the new user on the group. After the new path tree is selected, it is necessary to enforce it in the network. Thus, it is sent to the IPT component. Here, the IPT Controller will be required to send the necessary commands to the various IPT Node modules on the network and to the MTO, in order to successfully enforce the updates to the existing tree. This mechanism provides the architecture with ways to update existing sessions, avoiding severely disrupting the service delivery to previous users.

After the successful enforcement of the network, the NME sends to the SME the session modification answer. The SME will then send a SIP invite request to the new users. If they are multicast, the user’s terminal sends joins to the multicast group assigned by the IPT, while if the terminal’s connection to the LON is unicast type, a response is sent to the SME with the port number to which the multimedia contents should be streamed to the terminal. The SME will then trigger the update unicast ports event already explained in the section 3.6.1.

Upon completion, the new users are successfully included in the new group and are able to start the reception of the desired multimedia contents.

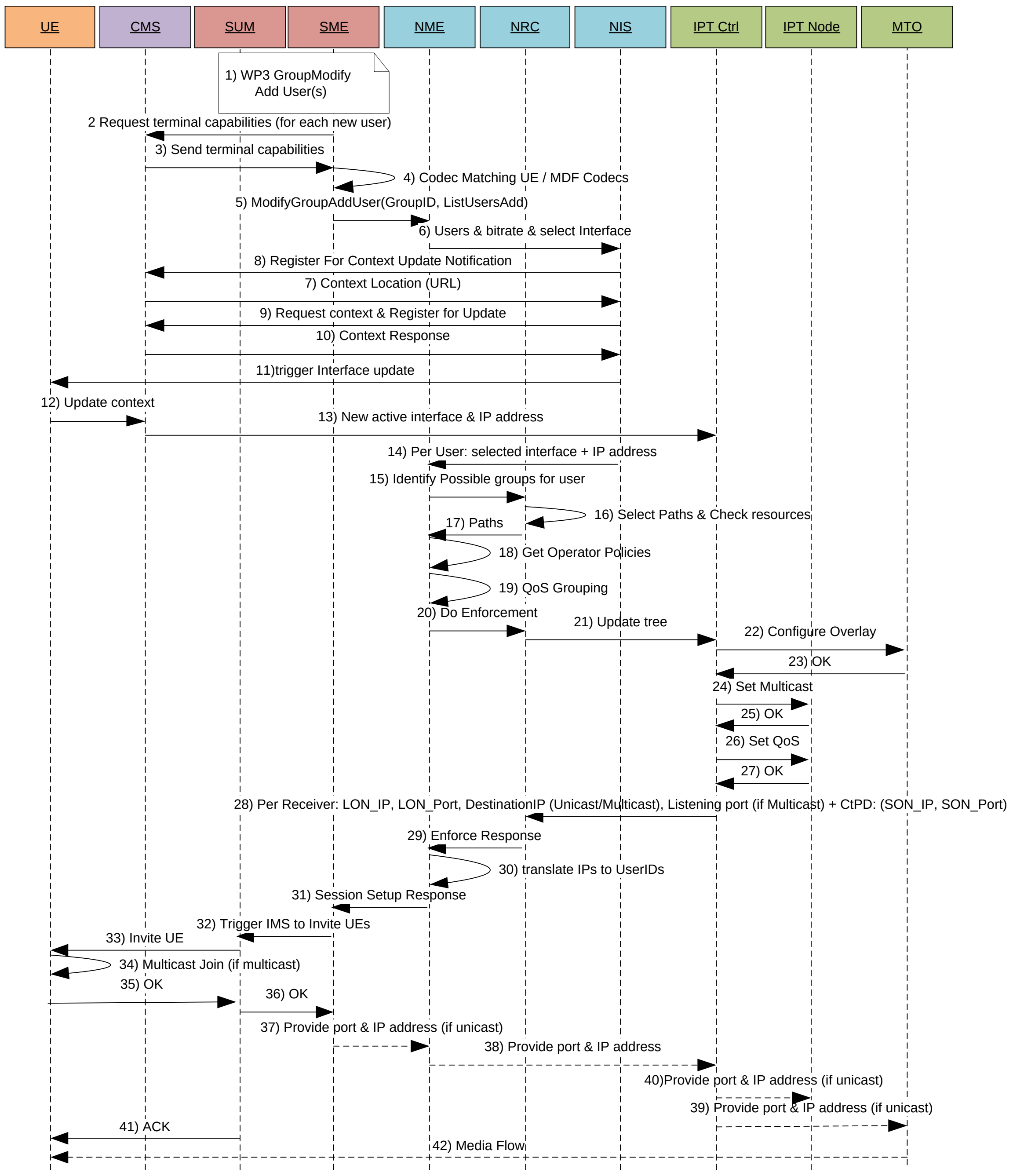

UE
CMS
SUM
SME
NME
NRC
NIS
IPT Ctrl
IPT Node
MTO
1) WP3 GroupModify Add User(s)
2 Request terminal capabilities (for each new user)
3) Send terminal capabilities
4) Codec Matching UE / MDF Codecs
5) ModifyGroupAddUser(GroupID, ListUsersAdd)
6) Users & bitrate & select Interface
8) Register For Context Update Notification
7) Context Location (URL)
9) Request context & Register for Update
10) Context Response
11)trigger Interface update
12) Update context
13) New active interface & IP address
14) Per User: selected interface + IP address
15) Identify Possible groups for user
16) Select Paths & Check resources
17) Paths
18) Get Operator Policies
19) QoS Grouping
20) Do Enforcement
21) Update tree
22) Configure Overlay
23) OK
24) Set Multicast
25) OK
26) Set QoS
27) OK
28) Per Receiver: LON_IP, LON_Port, DestinationIP (Unicast/Multicast), Listening port (if Multicast) + CtPD: (SON_IP, SON_Port)
29) Enforce Response
30) translate IPs to UserIDs
31) Session Setup Response
32) Trigger IMS to Invite UEs
33) Invite UE
34) Multicast Join (if multicast)
35) OK
36) OK
37) Provide port & IP address (if unicast)
38) Provide port & IP address
40)Provide port & IP address (if unicast)
39) Provide port & IP address (if unicast)
41) ACK
42) Media Flow

Figure 15 – Session Modification: New User message sequence [38]

3.6.2.2. Modified User

Modifications to existing sessions are not restricted to just the addition of new users; existing group users can also be modified (delivery path modified, connection type modified, LON modified, etc).

Existing users can trigger a modification event from two separate conditions. One of them is initiated by a successful network handover. In this case, the terminal's interface sends a SIP re-invite to SUM. A request to update the user is then sent by the SME to the NME. The second possible modification trigger is related to a change in the network, which is going to affect existing users or entire sessions. In this case, the NME triggers NUM to discover a new tree.

In both situations, the NME is required to send a session modification request to the IPT in order to enforce the updates on the network. The IPT Controller will then send the necessary commands to the various IPT Node modules and to the MTO.

After the enforcement on the network is complete, the NME will send to the SME the session modification answer, which will then perform a new set of SIP Invite requests to the new user's terminals. Afterwards, terminals will join to the multicast group (if they are connected through a multicast connection to the LON), or return to the SME the port number to which the multimedia contents should be streamed, triggering an update unicast ports event (if the terminals are connected through a unicast connection).

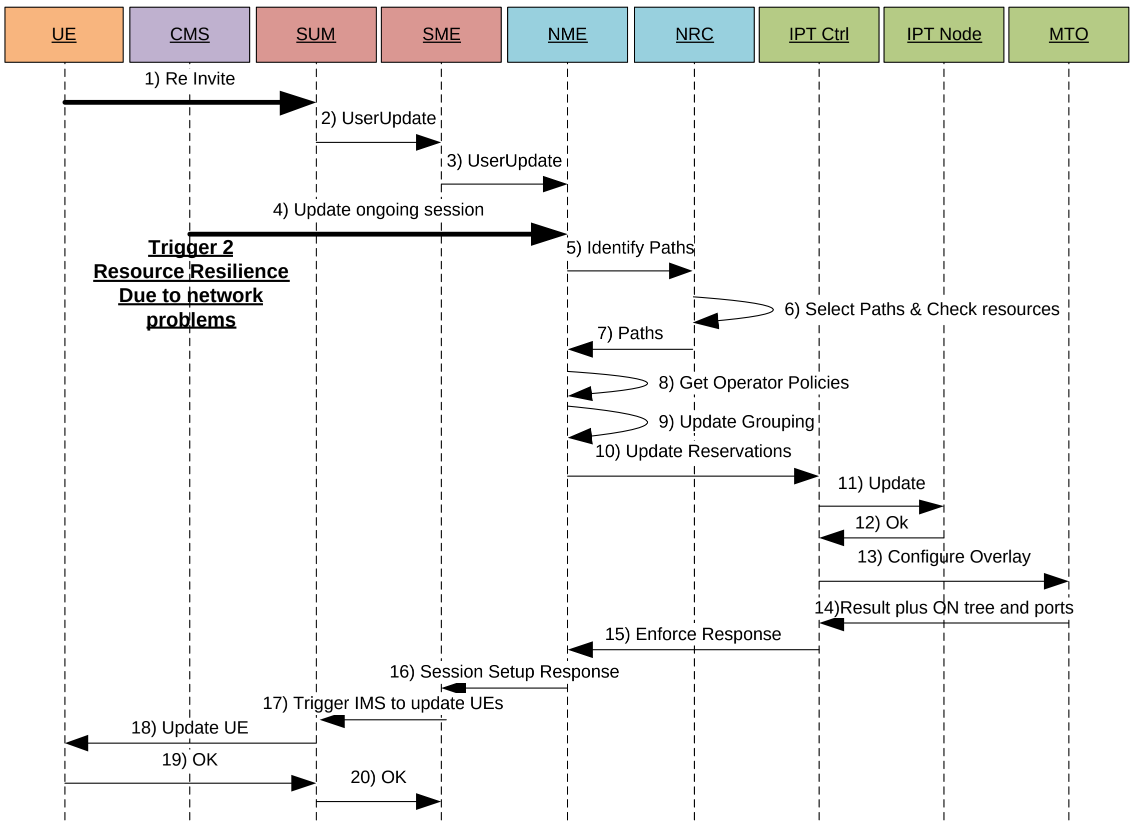


Figure 16 – Session Modification: Updated User message sequence [38]

3.6.2.3. Removed User

As already mentioned, users can also be removed from existing sessions. In this situation, the SME is triggered to start the removal of the user, sending afterwards a remove user request to the NME. Subsequently, the IPT is requested by the NME to remove the user connection from the LON. The path between the streamer and the LON (if no other user is receiving multimedia contents through that same path) should also be removed from the tree enforced in the network. These actions are performed by both the IPT Node module and the MTO.

When the update to the tree is complete, the NME returns to the SME the session modification answer. The SUM is then triggered by the SME to send a leave request (SIP BYE) to the user's terminals. If the user is connected through a unicast connection to the

LON, the removal is complete. However, if the connection between the terminal and the LON is multicast type, an additional leave request to the multicast group is necessary.

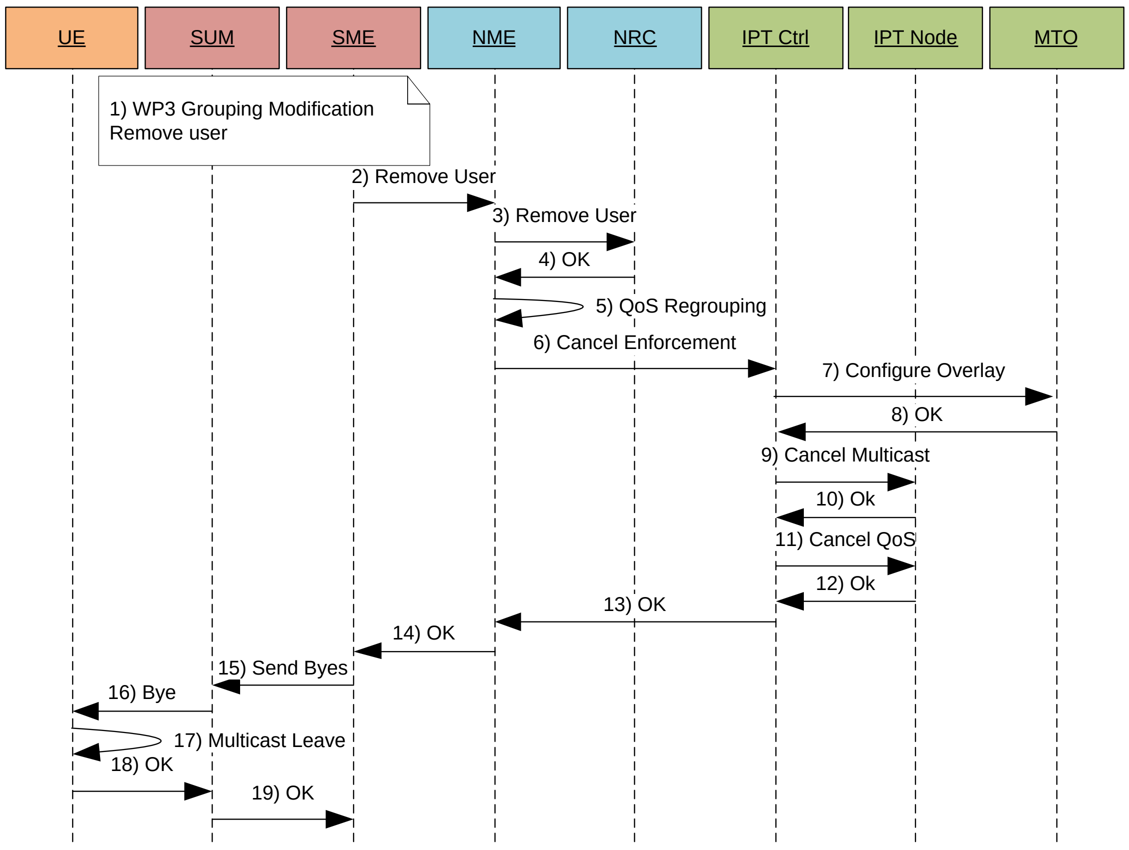


Figure 17 – Session Modification: Removed User message sequence [38]

3.6.3. Session Termination

The complete removal of entire sessions is also supported by the C-Cast architecture. This process is started by the SME, which is triggered to remove an existing session. The SME will then send a session removal request to the NME, which will trigger the IPT to release the resources allocated in the network for this session. Upon receiving this request, the IPT Controller sends messages to the various IPT Node and/or MTO to release the connections previously established.

After the removal of the tree is complete, the NME informs the SME that the session removal is complete. Thus, to finish this request, the SME triggers SUM to terminate both terminals and streamer, sending SIP BYE messages to them. The session is then completely removed, except for terminals connected through a multicast connection, which need to send an additional leave message to the multicast group.

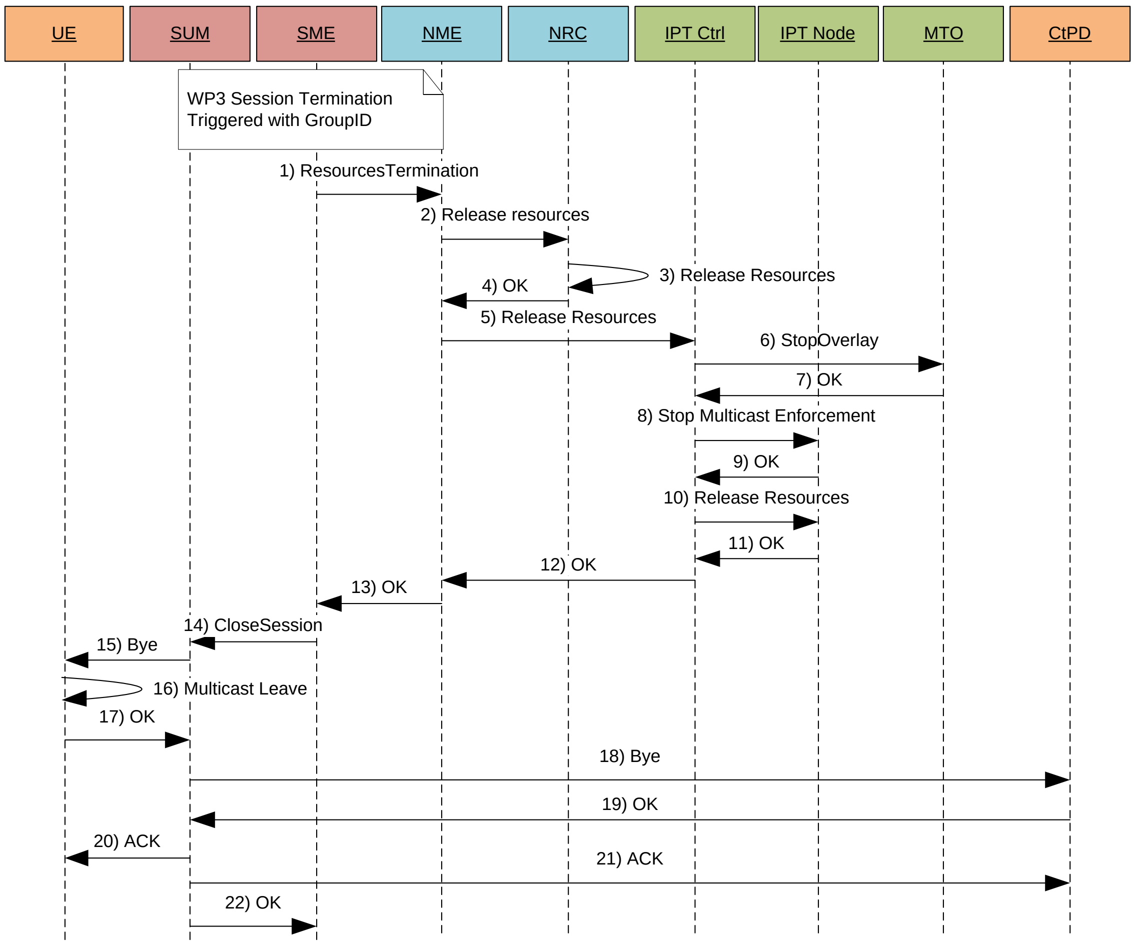


Figure 18 – Session Termination message sequence [38]

3.7. Summary

This chapter described the architecture proposed for the C-Cast project, the various components, as well as the interactions between them. The importance of the context-awareness was also explained, describing and explaining how its integration in the various elements in the network enabled them to dynamically adapt the transmission of multimedia contents to the groups of users. The integration of context information also allows a new level of management of the network resources, where both user and network context information is used to select the various formats, bitrate and core/access data paths for the multimedia content delivery.

On the other hand, various multimedia session events were also presented, which enable the interaction between several elements in the network to provide the selection of the more appropriate contents formats and bitrate values, the enforcement of the selected data paths on the network, and the necessary SIP signaling to allow the user's terminals to receive the multimedia data.

4. IP Transport

4.1. Organization

In the framework of this Dissertation, the IPT component of the C-Cast architecture and its interfaces were implemented and integrated in the overall C-Cast architecture.

Section 4.3 presents an overview of the intelligent central module of the hierarchical architecture proposed for the IPT. A detailed explanation of the created interfaces as well as the necessary internal processing is also provided.

Section 4.4 describes the complete implementation of the distributed module of the IPT's architecture, deployed in every router in the network, responsible for enforcing part of the IPT Controller decisions. An explanation of the mechanism used to apply and maintain the selected multicast groups on the network, as well as the procedure for the data exchange between the higher level and the lower level modules is also presented.

Finally, in Section 4.5 it is summarized the work done in this Dissertation to support the concept of dynamical networks driven by context.

4.2. Implementation Details

The IPT component proposed for this Dissertation was developed in the object oriented programming language C++. However, considering that the multicast daemon used (*mrouted*) had to be modified and since it was developed in C, part of the code developed for the IPT was in this language as well.

As for the development platform, the IPT component is deployed in a *Linux* machine (more specifically the Ubuntu 9.04). Considering the developed component, this machine is also required to have installed a Diameter stack (used for signalling between the IPT internal components) and the multicast daemon already mentioned (*mrouted*).

4.3. IPT Controller

The specified architecture for the IPT component can be described through a centralized and hierarchical approach, in which a single central module (IPT Controller) is responsible for all the high-level decisions and data processing. Connected to this central unit are various IPT Node modules, responsible for the enforcement of the multicast paths in the multiparty trees requested by NUM.

The IPT Controller, as mentioned, is responsible for processing the NUM requests and sending to the MTO Controller and the various IPT Node modules several instructions in order to successfully enforce the trees on the network. Therefore, it is essential that interfaces with the NUM, MTO and the various IPT Node modules are developed, in order to allow the tree requests to reach the IPT Controller and be enforced in the network.

4.3.1. NUM/NME Interface

The selection of the various paths in the network to be used to deliver the multimedia contents to the clients is done by the NUM component. It is also responsible for the detection of possible modifications in the network and triggering a session event to modify the existing enforcements. On the other hand, the NME is responsible for sub-grouping the various users by selecting the multimedia content files with the more appropriate bitrate values. Since the NME/NUM decisions are directly enforced by the IPT, it was necessary to develop a specific interface to handle the requests sent by this component.

This interface consists of several routines in the IPT Controller module that can be used in generic situations. Thus, when the NUM/NME is required to enforce a decision on the network, it selects the more appropriate routine provided by the IPT Controller. The request is then processed by the IPT Controller, ending in various instructions sent to the IPT Node modules and/or MTO to enforce the request on the network. To perform these tasks, however, the IPT Controller requires specific information relative to the decisions taken by the NME/NUM. Essential data such as a list of the nodes through which the multimedia contents should be streamed, the nodes to which user's terminals are connected and the various data paths selected, is necessary in order to enable a correct enforcement on the network. This data is then made available by the NME/NUM in a class named Multiparty Group upon the calling of the interface routines provided by the IPT Controller.

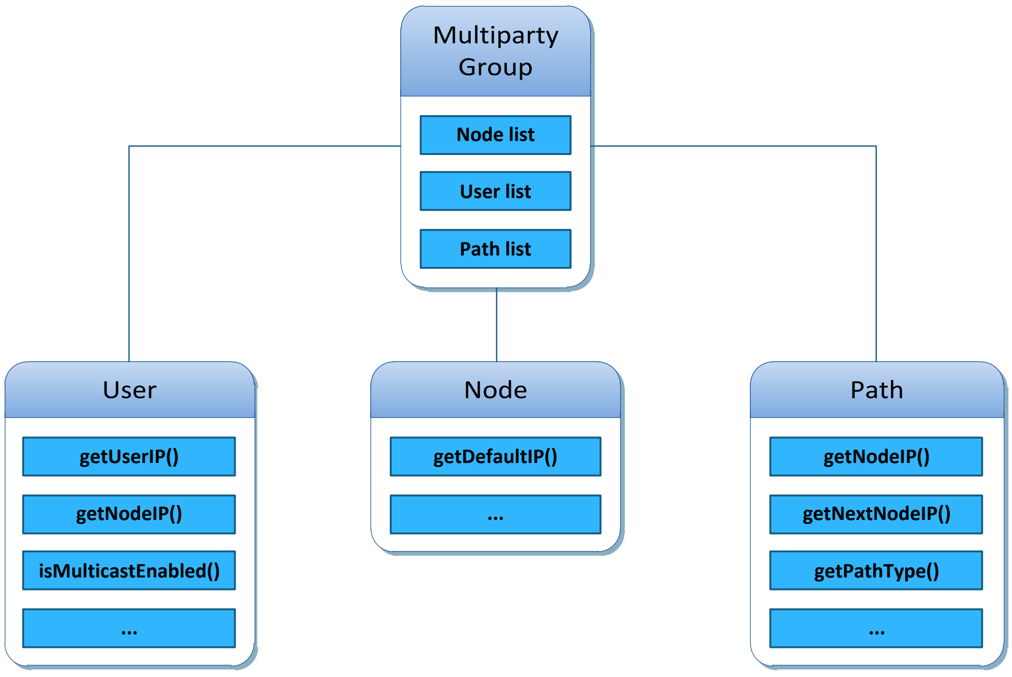


Figure 19 – Multiparty Group class and important functions associated

In this class, partially shown in Figure 19, some essential routines for the enforcement of the tree are presented. For example, when assigning a user to a stream, routines such as *getUserIP()* and *getNodeIP()* enable the IPT Controller to know what is the user's IP address and which is the node to which it is connected. Other routines such as *isMulticastEnabled()* inform if the user is going to receive the multimedia contents through multicast or unicast connection. Relatively to the paths and nodes, the routines *getNodeIP()* and *getNextNodeIP()* show which nodes connect each path, giving the IPT Controller a complete view of the tree setup. Another essential routine is the *getPathType()*, since it returns if that specific path supports IP multicast, or if it is just limited to unicast. Through this information, the IPT Controller is also able to decide if a specific network node should be enforced by the MTO or by an IPT Node module. This decision is based on the incoming and outgoing connections of the node. Since the IPT

Node module is only capable of enforcing multicast connections, if the node has at least one unicast connection, it has to be enforced by the MTO component.

The routines made available by the IPT Controller for this interface consist more specifically on the *Session_Setup()*, *Session_Remove()*, *Session_Modify()* and *Update_Unicast_Users()* routines. The first two are used, as the name suggests, for the creation and complete removal of a tree, while the *Session_Modify()* routine is used to modify paths or users in previously enforced tree. The last routine has a more specific usage as it is only used to update the port values of unicast users, and thus, more details for it will be given in section 4.3.1.2.

4.3.1.1. Session Setup

In the C-Cast architecture, users are grouped together due to similarities in their context information, and are registered to streams in order to receive the desired data. During the existence of a multimedia session, it is possible that some situations (increased/decreased wireless signal noise, handover to a new network with different characteristics, etc) change the user context information, forcing the NME to change the sub-group to which the user is registered. Sub-groups are a new concept introduced by the NME. As previously mentioned, the NME selects the more appropriate contents to be delivered to each user based on the bitrate value of those contents. The users are then placed in the same sub-groups if they are going to receive the multimedia contents with the same bitrate values. If the new sub-group to which the user is sent is empty (had no users before), the multimedia content stream specific to it is restarted from the beginning, forcing the user to receive media contents he already had received.

Thus, to avoid this problem, it was specified that every possible multimedia stream in the network (one for every possible sub-group) would start at the same time, meaning that when the tree for the first stream is created, a tree for every sub-group available is also created, even if it has no users registered to it. This way, when creating a new tree, the IPT has to consider all possible sub-groups, creating complete trees for the sub-groups that have unicast/multicast users associated with it, and empty trees for the ones with no users associated. The concept of empty tree applied here is also new and introduced by the IPT, consisting in a tree where only the first node in the network (Source Overlay Node) receives the multimedia data streamed through a unicast connection. After receiving the contents packets, it immediately discards them, preventing that they are propagated to other nodes in the network. Thus, by using this technique, users are able to switch sub-groups without disrupting the multimedia content stream received.

In light of the necessary modifications to the creation of a new tree, the Session Setup routine starts by accessing the multiparty class provided to check if the user list is empty. If the list is indeed empty, a specific function to create empty trees is invoked, sending the resulting message to the module responsible for its enforcement. Since the empty tree has only one node with one incoming unicast connection, then the module responsible for the enforcement of empty trees on the network is always the MTO.

Considering the other possibility, where the user list provided by the NME/NUM is not empty, the IPT Controller starts by going through every node in the multiparty class and classifying each one of them as a first node, middle node or last node. As the nodes get tagged, the message to the MTO is built and the data for the IPT Nodes is placed in a buffer. As seen in Figure 20, the information related to the first node in the tree is always sent to the MTO, since the incoming connection to the node is always unicast type. The classification of all nodes serves as a method to ease the creation of the message for the MTO, since the creation of the MTO message has very specific rules to follow (more details on section 4.3.2).

After going through every node in the list, the message for the MTO is complete, and as such, it is sent. The IPT Controller then starts a polling wait until the answer from the MTO is received. This answer message from the MTO is used as an acknowledge message, informing the IPT Controller if every action requested to the MTO was completed with success, as well as for displaying the port values chosen for every connection in the network (except for terminals port numbers). If an error was returned, the IPT Controller returns immediately to NUM with an error value. Otherwise, the IPT Controller places the port values decided by the MTO in the multiparty class used by

NUM. The multicast address assigned by the IPT for the multicast groups are also placed within this data class.

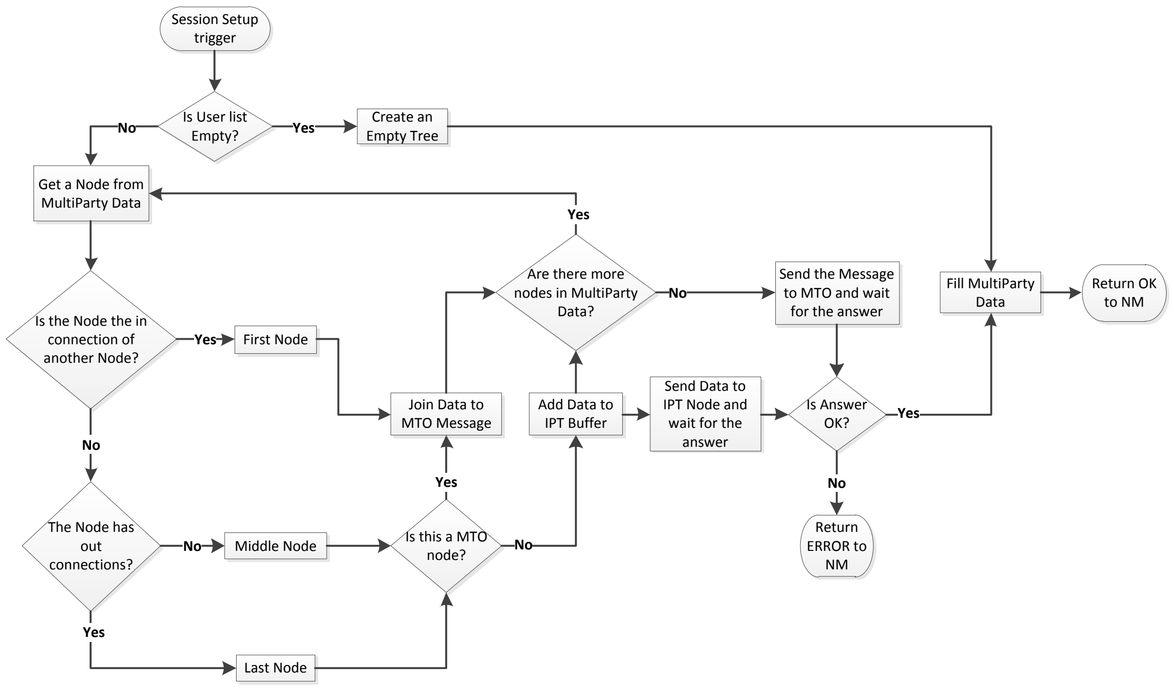


Figure 20 – Session Setup routine diagram

The control messages exchanged with the IPT Node modules have a completely different approach than those for the MTO. Instead of being grouped in a single message, they are split in several smaller messages, one for each interface to be added in each IPT Node. This way, when the IPT Controller decides that a specific node is being enforced by an IPT Node module, it sends one message for every interface to be added. Each of these messages contains the multicast group, the interface targeted and the message type (if it is a join or a leave message). Unlike the messages to the MTO, the messages for the IPT Nodes are not sent directly, but are instead placed first in a buffer. This buffer is checked periodically for new messages by a separate thread, and when new data is found, the thread builds several AVPs, placing them together in a single Diameter message. More details on the communications between the IPT Controller and the various IPT Node modules are given in 4.3.3.

4.3.1.2. Update Unicast Ports

After the trees requested by the NUM are successfully enforced in the network, the NME module returns to the SME. The SME then starts its SUM counterpart, initiating the SIP signalling required to assure that the multimedia contents are able to reach the user's terminals. The unicast terminals, upon receiving this message, return to SUM the port numbers to which the LONs should send the data packets. The multicast terminals however, do not return any information to the SUM. Instead, when informed by the SME of incoming streams, they just join the multicast group assigned to them by the IPT. Through this description, it is possible to conclude that, after the session setup event has been completed, the multicast terminals are able to receive the desired multimedia streams. However, the forwarding of the multimedia contents to the unicast terminals is not possible since the ports to which the streams should be sent is not yet known at the time the session is created.

To solve this problem, a new interface routine between the NME/NUM and the IPT Controller was developed. This routine is responsible for sending the updated information of the unicast terminals to the MTO module.

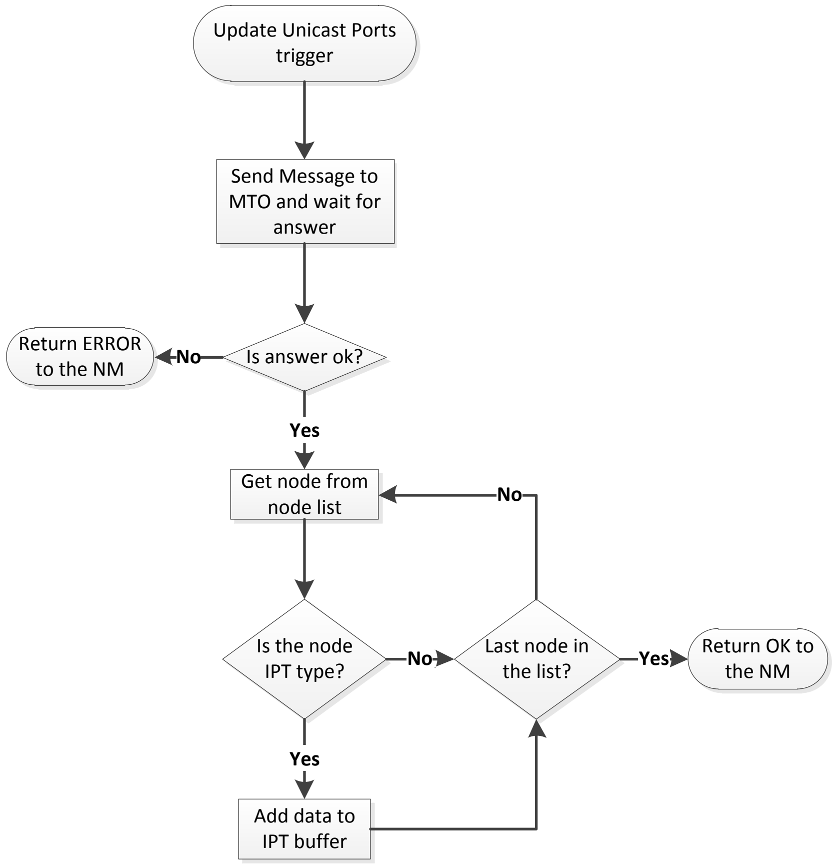


Figure 21 – *Update_Unicast_Ports()* routine diagram

The routine starts by searching every user in the data class provided by the NUM, selecting all terminals connected to the LONs through unicast connections and ignoring the ones with multicast connections. Then, the IPT Controller proceeds to discover which nodes the selected users are connected to, placing them in the MTO message being created. The rules on the structure of the MTO message prevent the same node to be listed more than once. Thus, before creating a new entry in the MTO message for the node to which the terminal is connected, the routine is forced to search in the existing MTO message to confirm if it was listed before. If it was, than the user has to be placed in that node. However, if no entry exists for this node, then it is necessary to create a new one.

Through this MTO message, the updated information on the unicast users is sent to the MTO module and applied to the network, allowing the LONs to correctly send the multimedia contents to the unicast terminals.

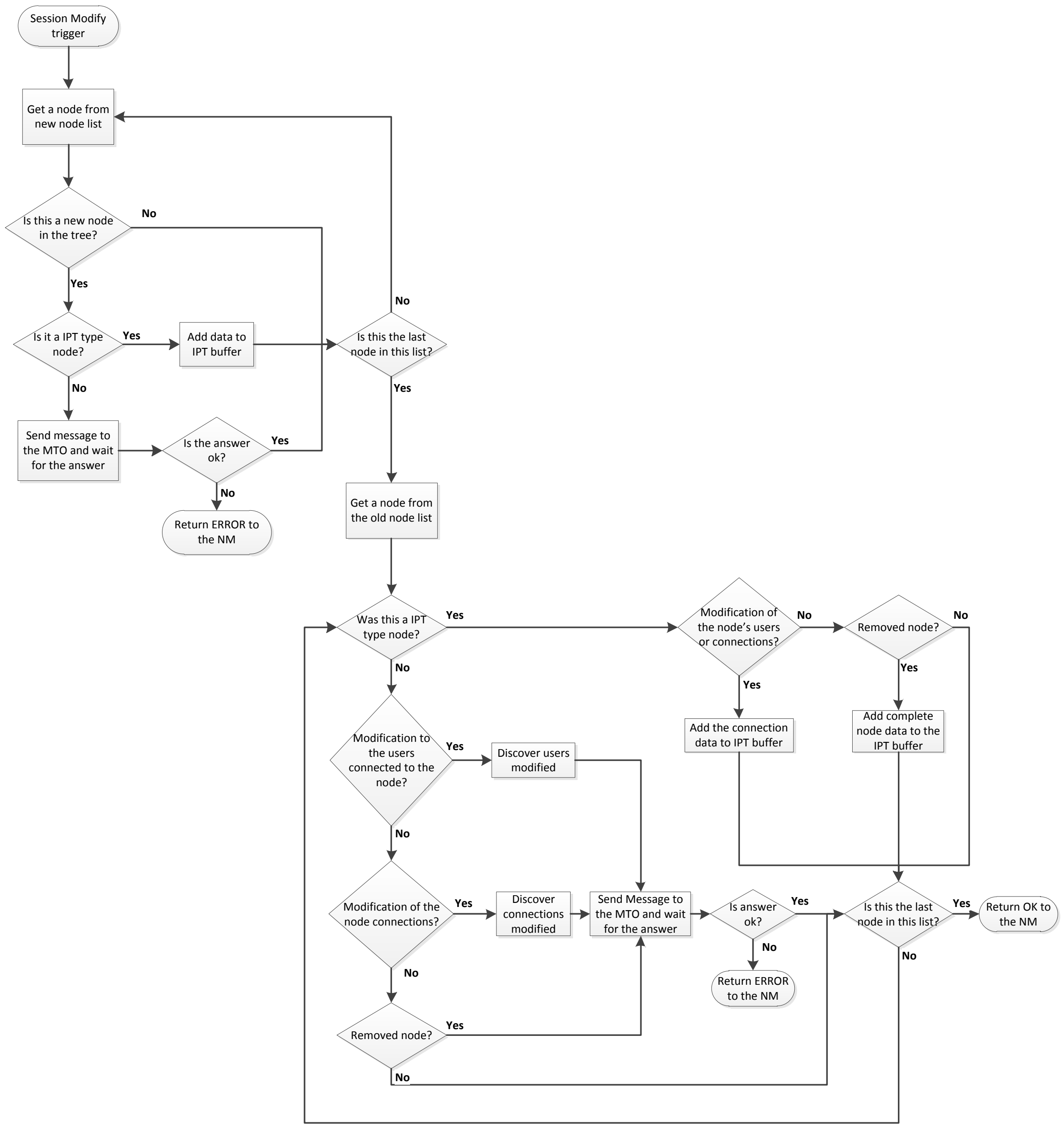
4.3.1.3. Session Modify

During the existence of a multimedia session, several modifications can occur. Users connected to a given LON may wish to stop the media contents being received. It is also possible that the node to which the terminals are connected changes, forcing NUM to select a new path between the SON and the new LON. On the other hand, even if the users are not directly subjected to modifications, the path through which the multimedia contents are being forwarded might be modified. This may happen due to changes in the network context, changing the ideal path in the core network. Situations where a link goes down are also possible, forcing a fast update to the existing tree. Without support for session modifications, these situations would force the IPT to completely remove the existing session and create a new one with all the modifications embedded. However, this would cause a complete service disruption for every user in the existing multimedia session. Thus, to support the modification of active sessions, a new routine was created in the IPT, the *Session_Modification()*. This routine is responsible for analysing the data provided by NUM, finding every modification to the existing session and taking the proper measures to ensure that the tree that is already enforced in the network is correctly updated.

The multiparty data provided by the NUM presents two lists for the node, users and path information. One list has the actual tree setup, while the other has the information of the previous tree setup. So, by comparing the new lists with the old lists, the IPT Controller is able to identify every modification to which the existing session was subjected.

The possible modifications an existing session is able to endure can greatly differ, even more if it is considered that modifications on some parts of the tree can cause modifications on other adjacent parts, considerably increasing the decision effort for the IPT Controller. Hence, it is essential that a predefined set of classifications is created, enabling each possible modifications to be split into several smaller modifications that are more generic and independent from each other. This mechanism allows the IPT Controller to correctly identify and process every possible modification to the network, no matter how complex, since they are always processed as a set of smaller and more manageable modifications.

The classifications are structured in a hierarchical way, with three major types describing the node type, followed by a set of smaller sub-categories which classify the possible modifications to which each node can be subjected.



Session Modify trigger
Get a node from new node list
Is this a new node in the tree?
No
Yes
Is it a IPT type node?
Yes
Add data to IPT buffer
No
Is this the last node in this list?
No
Yes
Send message to the MTO and wait for the answer
Is the answer ok?
Yes
No
Return ERROR to the NM
Get a node from the old node list
Was this a IPT type node?
Yes
No
Modification of the node's users or connections?
No
Yes
Removed node?
No
Yes
Add the connection data to IPT buffer
Add complete node data to the IPT buffer
Modification to the users connected to the node?
Yes
Discover users modified
No
Modification of the node connections?
Yes
Discover connections modified
Send Message to the MTO and wait for the answer
Is answer ok?
Yes
No
Return ERROR to the NM
Is this the last node in this list?
Yes
No
Return OK to the NM
Removed node?
Yes
No

Figure 22 – Session Modify routine diagram

The *Session_Modification()* routine (Figure 22) starts by going through every new and old node in the lists provided by NUM, classifying each node as an IPT or MTO type node. It can also be classified as a new node if the node is only present in the new node list. The first two classifications are made by searching through every connection associated with the node. If at least one of the connections is unicast, then the node is MTO type; otherwise, it is IPT type. These last classifications are done to decide which module should be responsible for the enforcement of the node modifications (if any) on the network. Considering the first two node types (IPT or MTO), after the node type is determined, the routine proceeds to find if the node was subjected to a modification. These modifications can be the addition/removal of users or paths connected to the node, the removal of the node or a type change (from IPT to MTO or vice-versa).

Despite the fact that all these modifications are possible on MTO and IPT type nodes, the information required by each module on the same modification is different. While the MTO requires detailed information about each node and which users or paths are connected to them, the IPT Node module simply requires the multicast groups each node's interface is connected to. For this reason, there is no difference between the path or user modifications to an IPT type node, since both correspond to an interface joining/leaving a multicast group. However, for the MTO type nodes, not only is a user modification completely different from a path modification, but even modifications to unicast terminals are different from the ones in multicast terminals. This last situation can be explained by the way users are assigned to the node by the MTO component. If the user's terminal is receiving the multimedia contents through a unicast connection, then it is created a new connection in the node to the specific user IP address. However, if the user's terminal receives the multimedia contents through a multicast connection, then a new connection is created from the node to the multicast group, and not to the specific user IP. Thus, it is irrelevant the number of multicast terminals that are connected to that specific node as long as at least one multicast user exists, since the multicast connection on the node should not be re-added if a multicast user is already connected to that node. In the same way, the multicast connection should not be removed if there are more multicast users connected to that node.

Other possible modification to the IPT or MTO type nodes is the presence in the old tree setup, while being absent in the new one, meaning that these nodes were removed from the new tree. Messages to the responsible modules for those nodes should then be sent, informing them that a complete removal of the nodes (as well as every user and/or path associated to them) is to be done. Since the paths created in the network during the session setup routine are bi-directional, then if a path between two nodes is removed, it has to be removed in both nodes. Thus, if a node is removed from the new tree setup, then it is also necessary to remove the path connections from every node adjacent to it.

For the nodes classified in the beginning as new nodes, the IPT Controller has to discover their type (IPT or MTO), using the same mechanism already stated here. After deciding which module is going to be responsible for the node's enforcement, the IPT Controller proceeds to send the appropriate message(s), successfully adding the node to the existing multimedia session.

After every node is classified and properly updated, the routine ends. The tree is then completely updated and the only service disruption experienced by the users is due to possible modifications in the nodes or paths leading directly to them, leaving intact (with no service disruption) the other users with no relation to the nodes/paths modified.

4.3.1.4. Session Remove

When every multimedia content in the list selected by the CSE has been streamed to the user's terminals, the session ends, and as such, its information needs to be removed from the nodes. A set of decisions is then taken by the SME and NME, ending in a session removal command for the IPT, with the purpose to completely remove the entire tree previously enforced. This request is sent to the IPT from the NUM component, invoking the *Session_Remove()* routine, responsible for sending the appropriate messages to the MTO and IPT Node modules to completely remove the existing tree.

The remove message to the MTO component only requires the ID of the tree to be removed (the MTO saves the tree setup associated with its ID upon its creation). After the message is sent, the IPT Controller waits for the MTO response confirming the successful tree removal. However, to completely remove the existing multimedia session, besides informing the MTO module, it is also necessary to inform the several IPT Node modules responsible for enforcing the remaining tree setup. Thus, the *Session_Remove()* routine searches every node in the class provided by the NM, selecting the ones that are IPT type and ignoring the ones MTO type. For each one of those selected, the routine proceeds to discover every interface that is assigned to a multicast group for the multimedia session to be removed. Then, a message is sent to each interface, specifying the multicast group to remove.

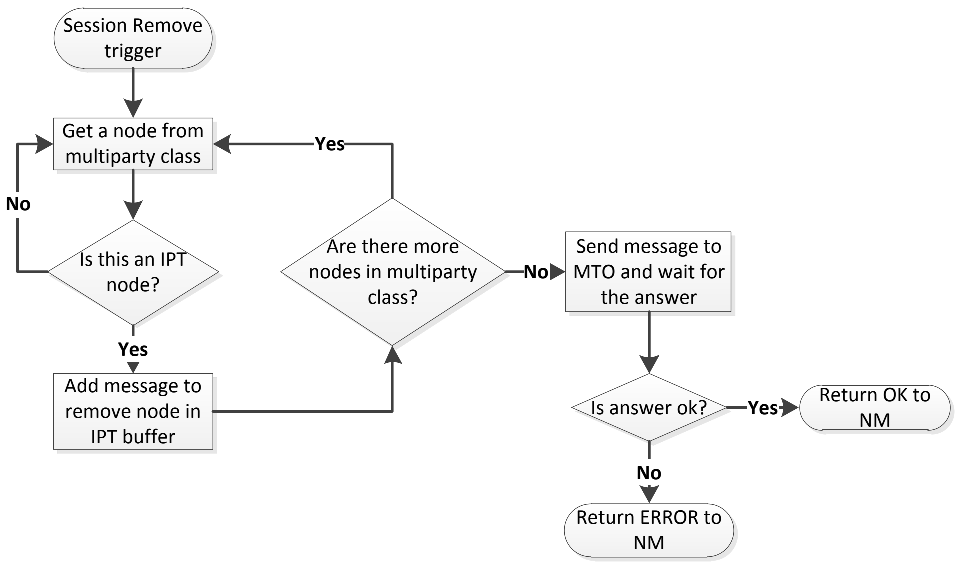

Session Remove trigger
Get a node from multiparty class
No
Is this an IPT node?
Yes
Add message to remove node in IPT buffer
Yes
Are there more nodes in multiparty class?
No
Send message to MTO and wait for the answer
Is answer ok?
Yes
Return OK to NM
No
Return ERROR to NM

Figure 23 – Session Remove routine diagram

After both removals are complete, the multimedia session is completely terminated and the IPT returns a successful message to the NUM component.

4.3.2. MTO Controller

As already mentioned, an interface between the IPT Controller and the MTO is required, in order to enforce the NUM requests on the network. This interface is based on the exchange of strictly defined control messages through a socket based connection.

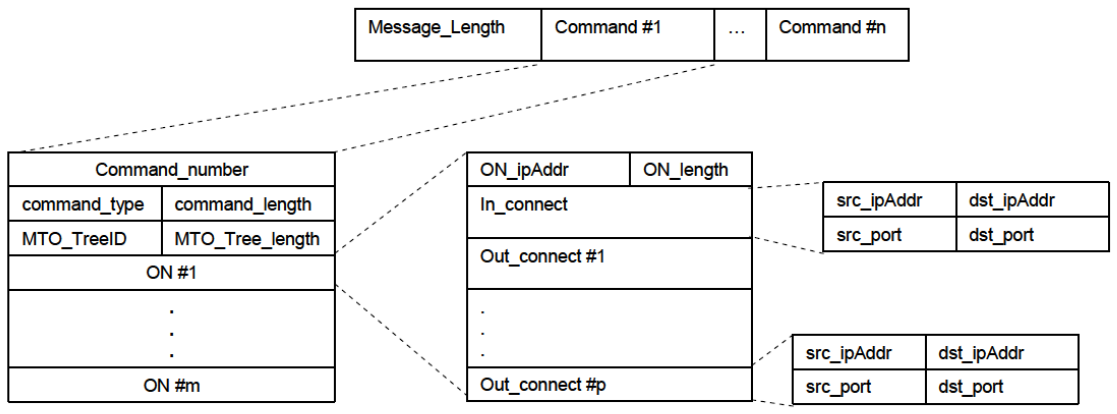


Figure 24 – Structure of the message sent to the MTO [38]

The structure of the message exchanged can be split in several smaller headers with different purposes. The command header contains the desired action to be taken by the MTO, as well as the number and length of the message. The command number field is used to clearly identify specific commands from each other, since a single MTO message can contain several commands at the same time. The tree header that follows is responsible for stating the tree ID to which the actions requested are to be taken. The last header is the one where all the important information related to the tree setup is placed. The data in this header is organized in a node list with a dynamic size, where each slot in the list contains several fields with information related to a specific node. In this list, all the information related to one node has to be placed in a single slot, meaning that the same node cannot be listed twice. Since the size is not static, the MTO has to read the value of the variable *MTO_Tree_length* to clearly know how many nodes are listed in the message.

Each node in the node list has specific fields that allow the MTO to identify the node's IP address, as well as every connection to and from the node. These connections are placed in a list with a dynamic size, specified in the *ON_length* field, and each slot in the list contains the information for the origin and destination of the connection, as well as the source and destination port number. However, with only this information the MTO is not able to identify which is the incoming connection to the node. To discover this, the MTO would be forced to analyse the origin and destination of each connection. However, since each node can only have a single incoming connection, it was specified that the first connection listed in the MTO message for each ON would be the incoming connection. With this technique, it is possible for the MTO to immediately identify every connection type in each node in the list, since the first connection listed is always the incoming connection, and the remaining ones are the outgoing connections.

Create_MT_Tree	Creates a completely new stream tree
Remove_MTO_Tree	Removes an entire tree already created
Add_Connection	Adds a new connection from a node to a user, multicast group or to another node in the tree
Remove_Connection	Removes a connection that already exists
Add_ON	Creates a new node to a session tree that has already been created
Remove_ON	Completely removes a node from an existing tree, as well as all its connections to nodes, users and multicast groups

Table 1 – MTO message commands

To perform the enforcements in the network requested by the IPT, the MTO provides a series of commands, listed in Table 1. Each one of those commands is used for a very specific set of actions in the MTO, and in each one, the message structure presented in Figure 24 has to be adapted to the requirements of the command.

When sending a *Create_MTO_tree* command, the rules followed by the IPT are generally the same as described above. The only differences are relative to connections to unicast and multicast terminals. Since the unicast terminals are not able to receive the stream right after the session setup (as explained in section 3.6.1), it was decided that the information for the unicast terminals would only be sent in a following session event named Update Unicast Ports, and as such, they are not included in the MTO message for this command. On the other hand, the multicast terminals information has to be included in this command.

To remove an already created session tree, the IPT should use the command *Remove_MTO_Tree*. This command differs completely from his opposite (*Create_MTO_Tree*), since no data relative to the tree is sent. This is due to the MTO storing the complete tree information and tagging it with a unique identification number. To completely remove a path tree, the IPT Controller should then leave the node list empty in the message to the MTO, placing only the identification number of the tree to be removed in the *Tree_ID* field.

While the two commands already mentioned are essential to create a complete new or remove an entire multimedia session tree, there are other situations, namely the session modifications, which require commands that can act on specific parts of the network without affecting the entire tree. One of these commands is the *Add_Connection*, created with the purpose of enabling the IPT to add connections to specific nodes. This feature is essential to the successful operation of the IPT module, since some of the routines in the interface between IPT and NME/NUM (e.g. *Update_Unicast_Ports()* routine) require that connections to unicast terminals are added to nodes that belong to an already created tree. In the message to send this command to the MTO, the IPT Controller is only required to fill the data relative to the new connections in the appropriate slot in the node list and specify which tree is going to be changed (tree ID). Since the MTO has completely stored the previous tree setup, the unmodified parts can just be ignored by the IPT and be left out of the message. The opposite situation, where a connection has to be removed, is done by the *Remove_Connection* command. Since the rules for the creation of these two messages are the same, the command distinction is just to enable the MTO to know if the connection should be added or removed.

The two last commands presented in the Table 1 are the *Add_ON* and *Remove_ON*, created with the purpose of enabling the IPT Controller to add or remove specific nodes on an already existing tree. These two commands, as the *Add_Connection* and *Remove_Connection*, are essential to provide the IPT with the capability of modifying trees that were previously created. The message for the first command (i.e. *Add_ON*) is created by specifying which nodes are new in the tree and including all connections associated with them. For the *Remove_ON* command however, it is only necessary to send the IP address of the node being removed, without including the node's connections. This characteristic is similar to the one in the command *Remove_MTO_Tree*, and is also due to the MTO saving the complete tree information at the time of its creation. For this reason, it is only necessary to provide to the MTO the ID of the node being removed (the node's IP address in this case) and the MTO will recognize every connection associated with it, removing them as well.

After completing the requested enforcements requested by the IPT in the network, the MTO needs to report to the IPT the success or failure of its actions. This is done through an acknowledge message sent in the reverse direction.

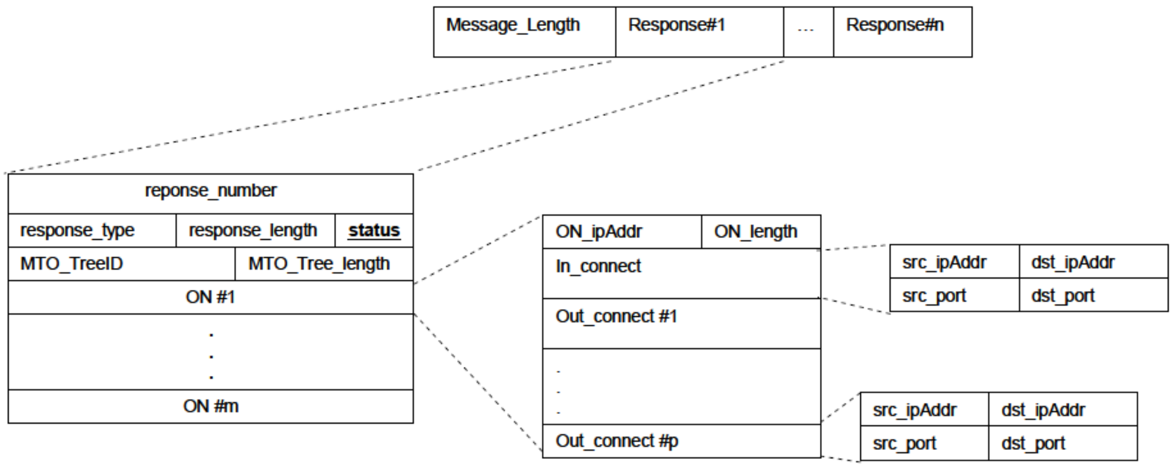


Figure 25 – Structure of the answer sent by the MTO [38]

This message, presented in Figure 25, is very similar to the one sent by the IPT. However, it contains an additional field in the command header, informing the status of the requested enforcements. If the status field is set to zero, then the MTO was able to correctly enforce the IPT requests. On the other hand, if instead of zero, the status field is set with any other value, then there was an error with the command requested, preventing the MTO to correctly finish the actions requested.

4.3.3. IPT Nodes

4.3.3.1. Start-up Registration

As explained, when the IPT receives the NUM requests, it enforces them in the network through the MTO and the various IPT Node modules. The interface with the MTO is done through the mechanism explained in section 4.3.2, where every message sent by the IPT Controller has the same destination, the MTO Controller's IP address. The IPT Node modules, however, are distributed through the network forcing the IPT Controller to send the requests to specific nodes. To accomplish this, the IPT Controller is required to have a complete knowledge of every active IPT Node and which are the IP addresses assigned to them.

Since the open-source Diameter stack used does not support peer-discovery, it was necessary to create it, through a registration mechanism developed in the IPT Node module. When each one of the IPT Node module starts, the registration method begins to establish a peer-to-peer connection to the IPT Controller. If the IPT Controller is available, the connection is created, providing a communication mechanism between the two modules. On the other hand, if the IPT Controller module is not available, the IPT Node blocks until a connection can be created. Next, the IPT Node will gather the address information of all of its IPv4 interfaces and place them in a message buffer. Since the communication between the IPT Controller and the IPT Nodes is done through a Diameter stack, the messages exchanged between the two modules have to follow a pre-determined structure, as explained in section 2.4.3. This way, the information relative to the node's interfaces is placed within a new AVP, which will then be inserted in the Diameter message and sent through the peer-to-peer connection created.

On the IPT Controller side, however, the registration method forces some modifications to the normal program flow. Since the IPT Controller is requested to maintain connections to more than one IPT Node module, the IPT Controller might need to forward data to multiple IPT Nodes at the same time. However, this would mean that the IPT Controller would have to communicate with each IPT Node one at a time. Thus, to develop a solution for this problem, several threads responsible for the communication with the various IPT Node modules were created. Hence, every time a new connection to an IPT Node is established, a new thread (named here "smaller thread") is created and assigned to that specific connection, assuring that connections to several IPT Nodes can be done at the same time. On the other hand, since new connections can arrive at any moment, it is necessary that another new thread (named here "main thread") is created to wait for new IPT Node modules.

On this new thread system, however, it is essential that the IPT Controller is able to target a specific smaller thread to send the control data to an IPT Node. On the other hand, it is also necessary that smaller threads have knowledge of when new data is available to be sent. To support these features, a new vector containing an array of IPT Data structures was created. The layout for this vector is shown in Figure 26.

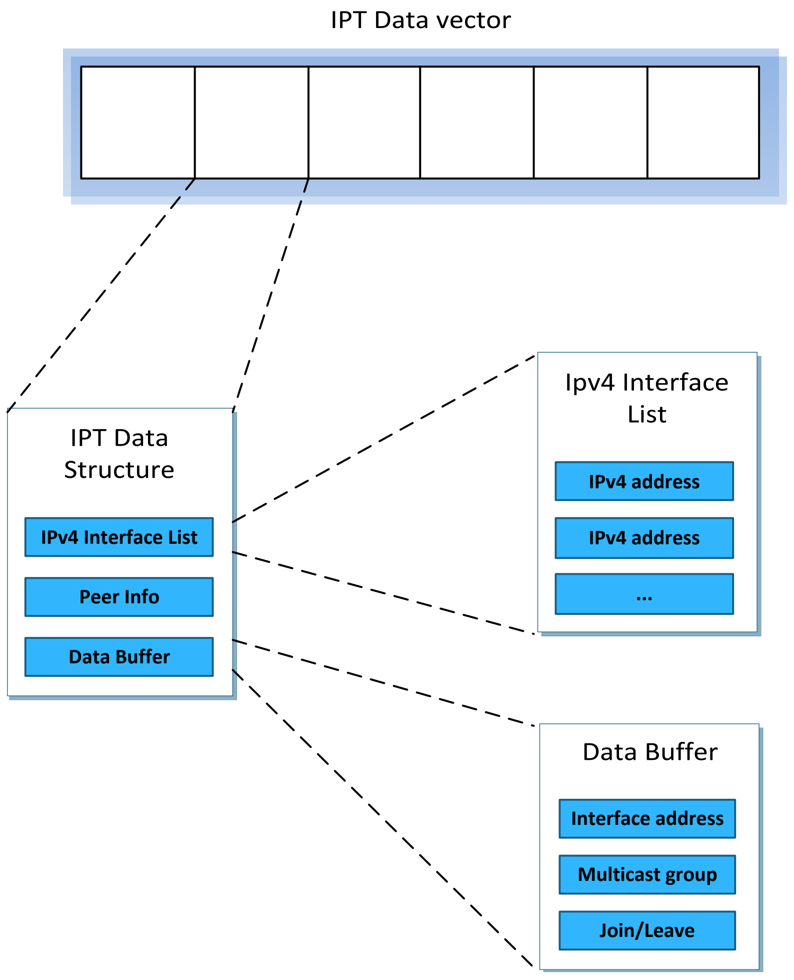


Figure 26 – IPT Data vector Structure

When a new connection is established, the main thread will stop and wait for the register message sent by the IPT Node. After the message is received, the IPv4 addresses sent by the IPT Node are placed in a new slot in the vector. Then, after the IPT Node is completely registered, a new smaller thread is going to be launched. However, since it is necessary that this new thread has access to the correct slot on the IPT Data array, the main thread passes the slot index for the vector as argument at the thread's creation. After this procedure is complete, the IPT Controller has gained a complete knowledge of the IPT Node interfaces and is able to send data to it through the use of the smaller threads.

4.3.3.2. Joining/Leaving multicast group

In the last section (4.3.3.1), it was explained the behaviour of the mechanism that is used by the IPT Controller to know which IPT Node modules are active in the network and what are their interface addresses. It was also explained how it is able to manage multiple connections to these modules. However, how the data is actually sent to the various IPT Node, and what is placed in this control message is still unclear.

As explained, each smaller thread has access to a specific slot in an IPT Data vector. In this slot, the thread has stored the information of the peer it is connected to and its IPv4 interfaces. Despite the fact that all this information is essential to the correct management of the existing connections, it is not enough to allow the IPT Controller to pass the information to a specific smaller thread. To allow this, an additional field behaving as a data buffer was added to the IPT Data structure. This new field acts as a shared buffer between the IPT Controller and the targeted smaller thread, enabling the IPT Controller to send data to a specific smaller thread. This way, when during one of the routines used in the interface with the NME/NUM the IPT Controller is required to send enforcements to the IPT Nodes, all it needs to do is to insert the message to be sent in the correct IPT Data slot. Since due to the data provided by the NUM module, the IPT Controller has knowledge of the IPv4 address of the node targeted, discovering the correct position in the IPT Data vector is done through a search for the vector's index that has the desired IPv4 interface. After the correct index is found and the data is inserted in the correct buffer, the IPT Controller's routine can continue.

Thus, the smaller threads are only required to check the data buffer on the correct vector index. If the buffer is not empty, then the IPT Controller has placed data that has to be sent. Hence, the data is removed and transformed in several AVPs.

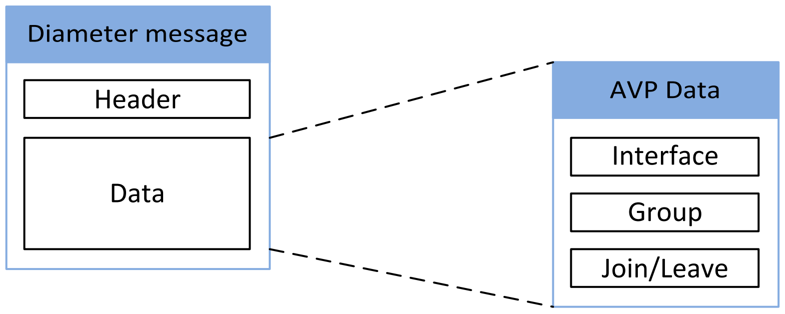


Figure 27 – Diameter message with AVPs sent to IPT Nodes

Since the data is divided in three essential fields, the smaller thread is required to create three different AVPs. The first contains the node's interface being targeted, while the multicast to join/leave is sent in the second AVP. The third AVP contains the message type (i.e. if the interface is requested to join or leave the multicast group). The three AVPs are then combined in a single Diameter message (Figure 27) and sent through the peer-to-peer connection previously established.

It is important to notice that, since the data buffer present in the IPT Data structure can be accessed by two different processes at the same time, it is essential that the data is protected from simultaneous usage. Thus, a *mutex* variable was created to prevent simultaneous accesses to the data buffer. However, since the smaller thread is checking the data buffer in a continue loop, starvation problems might appear on the IPT Controller, completely preventing it to access data in the buffer. To solve this issue, a small wait was inserted between two consecutive accesses to the data buffer, giving the IPT Controller a much more manageable time window to access it.

4.4. IPT Node

The IPT Node module is distributed through every node in the network, receiving the enforcement requests from a central component, the IPT Controller. The task endorsed by this module is to receive, through a peer-to-peer connection created with the IPT Controller, the requested commands in a Diameter control message.

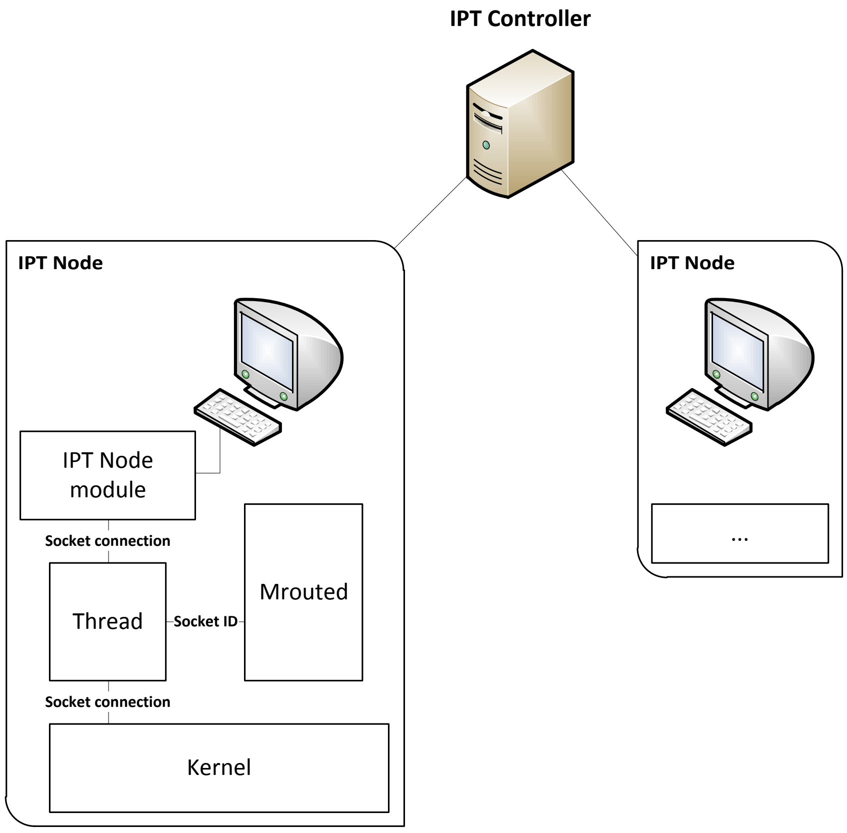


Figure 28 – IPT Node architecture

As explained in section 4.3.3.1, as soon as the IPT Node module starts, a connection to the centralized unit (IPT Controller) is established using a Diameter stack. Next, the information of every IPv4 interface is gathered and placed within a vector. The vector is then used to create an AVP, which is then inserted in a Diameter message and sent to the IPT Controller. After this procedure, the node is properly registered and is able to receive the IPT Controller requests through the same peer-to-peer connection.

The requests sent by the IPT Controller are placed in a Diameter message, depicted in Figure 27. Inside the data field of this message, three AVPs are contained, instructing the IPT Nodes of which is the interface and multicast group that should be targeted, and if this is a join or a leave message. After the AVPs are correctly de-encapsulated, it is necessary to instruct the machine's kernel to perform the desired action.

To support the IP multicast routing, a multicast daemon based in DVMRP was installed, acting between the IPT Node module and the machine's kernel. To add or remove a multicast group from a specific interface, the IPT Node is required to create a new socket and modify its options, such as the incoming and outgoing addresses, to match the IPT Controller's request. Since new requests can arrive to the IPT Node at any time, it is necessary that a new thread is created. Thus, while the IPT Node is responsible for listening for incoming control messages, the new thread is responsible for creating and managing the multicast sockets. The management of the multicast control messages exchanged between neighbour nodes is then performed by the multicast daemon *mrouted*.

4.5. Summary

This chapter presented a complete overview of the implemented solution for the IPT component.

Despite the challenge and difficulties associated with its implementation, it is possible to conclude that the specified features and functionalities were successfully achieved.

The central module implemented in the IPT architecture, named IPT Controller, is responsible for identifying the NUM requests and correctly enforce them in the network through the various IPT Node modules and the MTO. As such, the development of an interface with all these components was necessary. Details of the interfaces created were also given, where the proprieties of the control messages exchanged are explained.

The implementation of the IPT Node module, deployed in the network, was also exposed in detail, carefully explaining how the interaction with the central module (i.e. IPT Controller) was developed. It was also explained how the specific multicast paths determined by the IPT Controller were enforced and maintained in the network.

5. Results

5.1. Organization

This chapter presents a performance evaluation of the proposed architecture based on the results from different scenarios. These scenarios were chosen in order to provide an overview of the implemented features and functionalities, and their impact on the network.

In section 5.2 contains information on the development of the IPT component, explaining how the modules were integrated and tested in cooperation with the remaining C-Cast architecture elements.

Section 5.3 presents various test scenarios with the objective of presenting an example of the IPT behaviour during various session events. In these tests, the messages exchanged by the IPT are also presented.

Section 5.4 provides an evaluation of the message overhead created in the network by the IPT. Several scenarios were also studied in order to identify the major factors that influence the amount of control data exchanged.

Section 5.5 presents the necessary processing time of IPT in various session events, considering for each one different situations. The processing times are then compared with the components directly involved in the enforcement of the trees on the network. Finally, it is performed an evaluation of the processing time required by the IPT and the multiparty transport framework in comparison with the overall time required for a session event.

Section 5.6 presents a summary of the performance results, giving a final evaluation of the performance observed for the developed component.

5.2. Development and Integration

The objective of this dissertation was to specify and develop the IPT component, and at the same time integrate it in the proposed architecture for the C-CAST project. Thus, during the creation of the module, performing internal tests on the component was not enough. It was also necessary to integrate the module developed on various phases with the remaining architecture components. The development of the IPT module can be split in three different phases.

The first phase was the creation of an interface with both NUM and MTO to enable the creation and removal of trees. During this time, the tests were performed with simple network setups, since the support for the session modification event was not implemented, as well as the support for the multicast enforcement by the IPT Nodes. Thus, simple trees were emulated in NUM and sent to the IPT, where they were processed. The IPT would then send messages to the MTO containing the necessary actions to be taken in order to enforce the requested tree in the network. The session removal event was also tested in this phase, where the NUM instructed the IPT Controller to remove the previously enforced tree on the network.

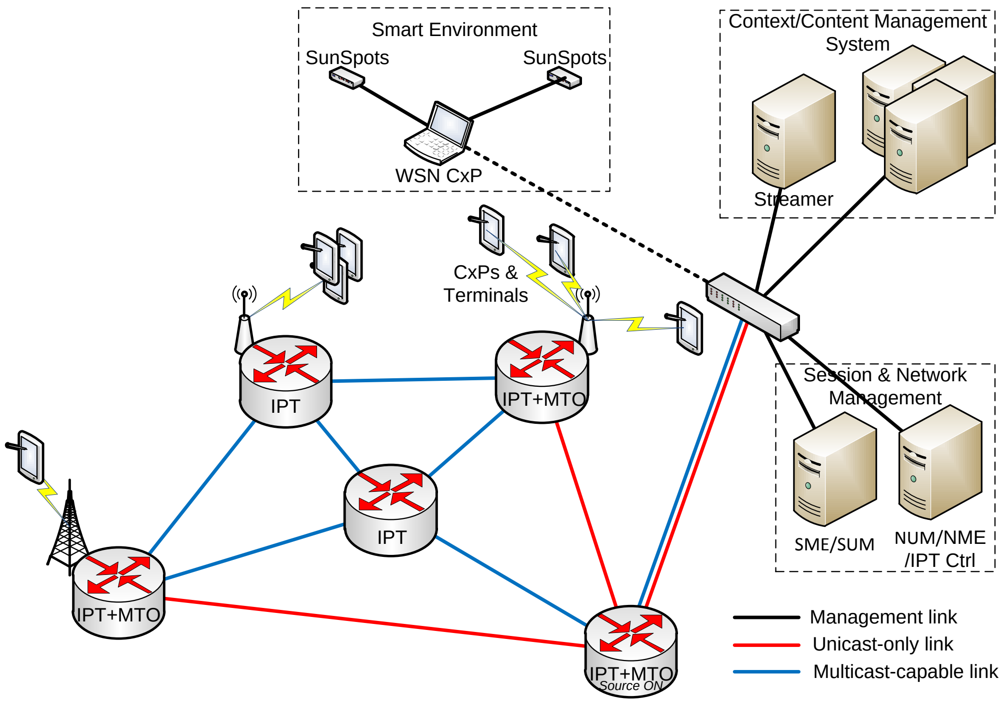


Figure 29 – Testbed scheme for the various tests performed

The second phase was the development of the support for the session modification event. This new event type had a completely new set of tests, since the network was now supposed to be able to react to modifications such as users changing the nodes they were connected to, users leaving the user group and new users joining. Thus, several scenarios were created and tested thoughtfully, both with the IPT alone and with the remaining elements of the architecture.

The last phase of development was the integration of the support for the enforcement of multicast groups along the nodes where the IPT Node module was deployed. In this last phase, multiple tests were completed with just the IPT, where the communication between the central module (IPT Controller) and the distributed IPT Node modules was completely tested. In these tests, the successful creation and removal of multicast groups in specific paths in the network was also tested. Furthermore, after the basic tests were performed and successfully completed, this last new feature was included in more tests with the remaining network elements in the C-CAST architecture. In these tests, the multicast support for the session creation, modification and removal was tested, guaranteeing the successful enforcement of the required multicast groups along specific paths in the network.

5.3. Test Scenarios

In this section, various tests are presented with the IPT component. First, a new multimedia session is going to be initiated, with only one unicast user. Then, this user is going to be subjected to a context change, forcing an update on the network through the session modify event. After the existing user is correctly updated, a new multicast user is going to join the same user group, triggering a session modification event as well. Finally, the complete multimedia session is going to be removed through the session remove event. In all these tests an explanation of the necessary interactions between the various architecture components is going to be given, as well as examples for the signalling messages exchanged between the IPT Controller and the IPT Node and MTO.

The first test is going to be performed with the network setup presented in Figure 30. It is also going to be considered that the unicast user is going to watch a movie, and as such, it will receive two separate streams (one for audio and one for video). As it can be seen, only one user exists in the user group and it is connected to node 5 through a unicast connection. Since every paths in the network through which the multimedia contents should be forwarded are all unicast type, then the complete enforcement of this tree is going to be done by the MTO component.

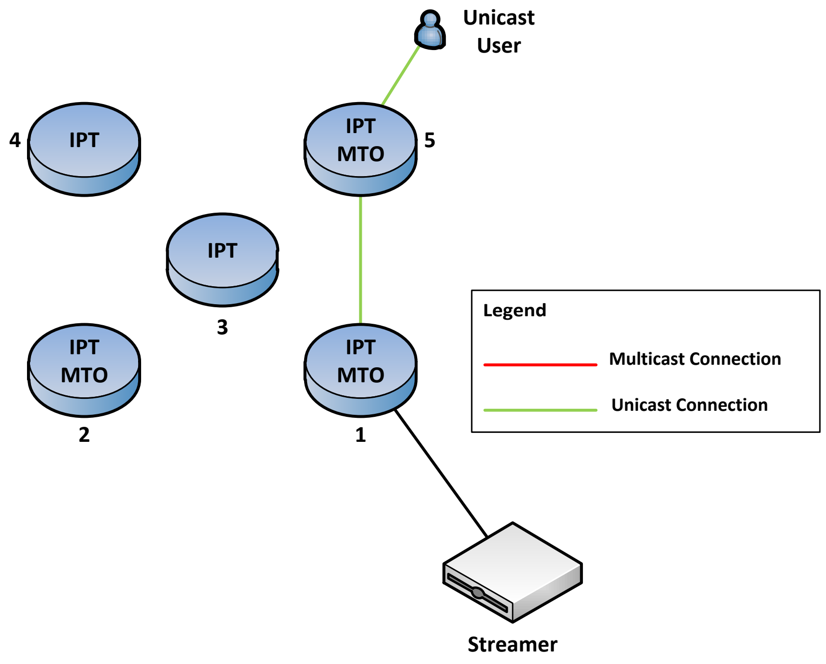


Figure 30 – Scenario for the first test

Thus, in this scenario the IPT Controller will act as an interface for the NUM requests, processing and organizing them into a single message that the MTO is able to understand. Upon the successful creation of the user group by the GME module and the content selection by the CSE component, the SME is going to be triggered to start a new session. Here, the more appropriate codecs for the multimedia contents are going to be selected, based on the user's capabilities and preferences. The list of the supported/preferred codecs is then sent to the NME, which will select the file to be streamed based on the network situation and the file's bitrate. The NUM is then invoked to select the best path from the streamer to the user and send to the IPT Controller the selected network setup. The IPT Controller will then send a *Create_Mto_Tree* message in which the information of every node and connection essential to enable the delivery of the multimedia contents to the user is going to be sent to the MTO (Table 2). It is also important to notice in the message shown that, in the second ON included in the MTO message, only the information relative to the incoming connection is present, while the information for the connection between the node and the user's terminal is not present. As explained, unicast users are not able to receive the streams after the creation of a new session; it is first necessary to update their ports. Thus, the information for connection between node 5 and the terminal is only going to be sent in the following *Update_Unicast_Ports* command. As a side note on the command message presented, the real MTO message would have the node IP addresses instead of the node number shown. However, for simplification purposes, the node's IP addresses are going to be replaced by the node number.

Type:	*Create_Mto_Tree*	
ON -> 1		
	Src -> Streamer	Dst -> 1
	Src -> 1	Dst -> 5
ON -> 5		
	Src ->1	Dst -> 5

Table 2 – MTO message for first test

Considering that the user is receiving two different streams, the message presented in Table 2 is going to be sent twice, each one with a different tree ID.

After the tree is successfully enforced in the network, the MTO will return to the IPT Controller an acknowledge message, informing of the success of the requested actions. The IPT will then return to NUM with success, which by its turn will inform the SME of the session status. The SME will then perform the SIP signalling messages, inviting the unicast user to the multimedia session. The terminal will then inform the SME of the port number that should be used by node 5o stream the multimedia contents. In this situation, a new event is going to be triggered by the SME component, the Update Unicast Ports. This event is then sent to the NME. Upon the reception of this request, the IPT Controller will send to the MTO the control message shown in Table 3. This message includes not only the details on the connection between node 5 and the terminal, but also the details on the port numbers that should be used to forward the multimedia contents.

Type:	*Add_Connection*	
ON -> 5		
	Src ->5	Dst -> Unicast User

Table 3 – Second MTO message for first test

After the enforcement of this connection on the network, the user is able to receive the incoming multimedia streams, and as such, the session was successfully created.

In the second test, it is going to be considered that the existing unicast user is going to move to a significantly noisy location. This modification is detected by the CxPs, which will inform the CxB of the context modification. The SME component is then noticed of the user context modification. Considering that the unicast user was watching a movie (and as such receiving two streams, one video and one audio), the SME is going to decide that the noise modification is degrading the user's QoE and as such, it is going to replace it by the same video but subtitled instead. To perform this modification, the SME is going to trigger NME to remove the existing streams and create a new one. NUM will then select the best paths in the network and send the complete tree setup to the IPT Controller. Here, three control messages are going to be created: two *Remove_MTO_Tree* commands for the existing streams and one *Create_Mto_Tree* for the new stream. The *Remove_Mto_Tree* does not contain any ON details, since the MTO only requires that the tree ID is specified. The *Create_Mto_Tree* however has to be completely specified, as it can be seen in Table 4.

Upon the successful enforcement on the network, the MTO will return an acknowledge message to the IPT, which will inform NUM of the success of the enforcement on the network of the modification to the multimedia session.

Type:	*Create_Mto_Tree*	
ON -> 1		
	Src -> Streamer	Dst -> 1
	Src -> 1	Dst -> 5
ON -> 5		
	Src ->1	Dst -> 5

Table 4 – MTO message for second test

For the third test, it is going to be considered that a new user is going to be placed in the existing user group. The new network setup can be seen in Figure 31. It is important to notice that this new multicast user is connected to a different node than the existing unicast user. Also, considering that nodes 3 and 4 only have multicast connections, these are going to be enforced by the IPT Node module instead.

This modification to the existing multimedia session starts in SME, which is informed that a new user has joined the existing group. The more appropriate codecs are then going to be selected for the new user and sent to the NME, which will select the file with the best bitrate, considering the network situation. After the multimedia file is selected, NUM is triggered to select the new path through which the contents should be forwarded in the network. Upon selecting the new tree setup, it is sent to the IPT Controller. Here, the tree setup is going to be analysed in order to select the module that is going to be responsible for the enforcement of the new data paths. Considering that the new nodes only have multicast connections, the IPT Controller is going to decide that these modifications are going to be enforced by the IPT Node modules on node 3 and 4. The IPT Controller will then have to send one message for each IPT Node module, specifying every interface in the node that is going to join a multicast group, and the IP address of the multicast group. It also has to specify in these messages that the interfaces are going to perform a join request and not a leave request. After the modifications are enforced in the network, the IPT will return to NUM, and the SME is going to be informed of the new status of the multimedia session. It will then send a SIP invite to the new terminal, which will then also join the multicast group. At this moment, the new multicast user is able to receive the multimedia contents, completing the modification to the existing tree.

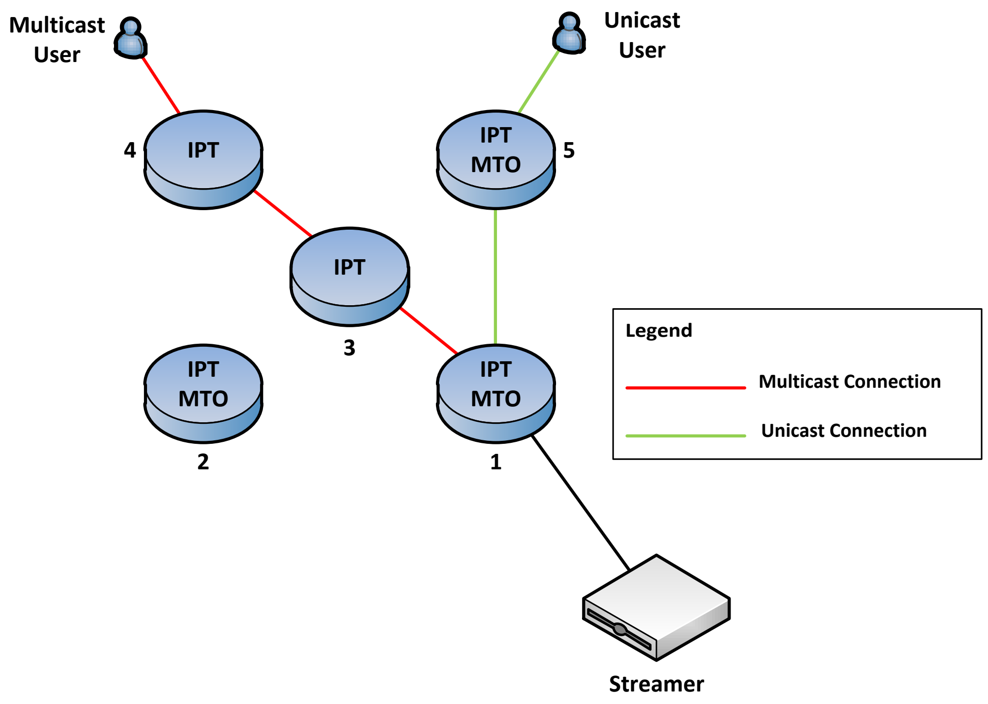


Figure 31 – Scenario for the third test

Still considering this session modification event, it is possible to verify that the existing enforcements on the network were updated through messages to nodes that were not directly involved in the delivery of the multimedia contents to the user who was already connected. Thus, the new enforcements on the network did not create any modification on the enforcements for the previous unicast user. Thus, it is possible to confirm that the IPT does provide mechanisms to successfully update the network's enforcements to match the modifications to which the session was subjected, while at the same time, these modifications do not create service disruptions to users who were not affected by these modifications.

In the final test, the existing multimedia session is going to be completely removed. Thus, the SME will trigger NME to perform a session remove event. NUM will then send the session remove request to the IPT Controller, which will inform the MTO and the IPT Node modules on node 3 and 4 that the current enforcements have to be released. The IPT Controller is only required to send a *Remove_Mto_Tree* message to the

MTO, specifying the tree ID to remove. For the IPT Node modules however, new messages have to be sent, specifying every interface that is going to leave the multicast groups. Upon the successful release of the current enforcements, the IPT will return to NUM, reporting the success of the requested action. The SME will then receive the session response from NME and will send SIP BYE messages for the two user terminals, successfully terminating the existing session.

5.4. Control Message Overhead Evaluation

In this section, it is performed an evaluation of the IPT performance considering the signalling messages. Various session events are going to be studied, where in each one, an evaluation of the impact of the IPT in the network as well as a comparison with other architecture components is described.

The first event evaluated is the session creation with a single user in the group. The results obtained for this test are presented in Table 5.

From – To	Signalling Message Size (bytes)
CDE – SME	1296
SME – CtPD	1739
CxB – SME	2931
SME – NME	3540
IPT – MTO	1257
SME – Streamer	3160
SME – Terminal	3613

Table 5 – Signalling Overhead evaluation for session creation

Through these results, it is possible to observe that the majority of the control messages are exchanged between the SME component, either to get the content list from CtPD and user context information from CxB, or to manage the SIP messages with the streamer and the terminal. In this session event, the messages exchanged between the IPT and the MTO (MTO message and acknowledge message) make up a small part of the total signalling messages exchanged. Furthermore, it is important to notice that this value includes both components together.

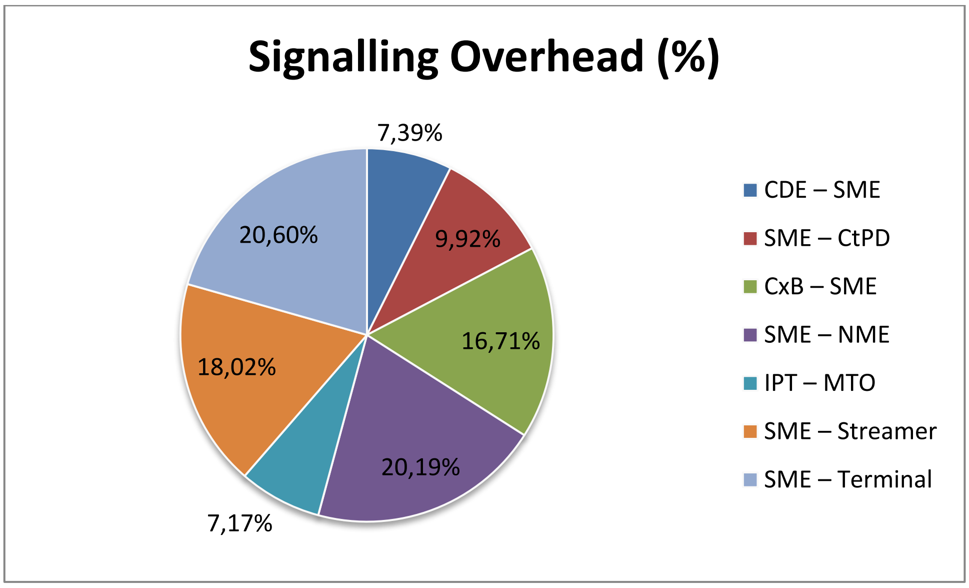


Figure 32 – Signalling overhead percentage for session creation event

In Figure 32it is shown that the signalling overhead for the messages exchanged between the IPT and MTO is approximately 7%. Since this value considers both the messages sent by the IPT and by the MTO, and knowing that these messages are approximately equal (they only differ on one field, the status of the actions requested), it is possible to conclude that the messages sent by each component only make up for 3.5% of the total signalling overhead on the network for the session creation event.

For a second test, a session modification event is going to be performed. In this event, one of the existing users is going to have its context information modified, and as such, it is necessary to update its enforcement on the network. The results obtained for this scenario are presented in Table 6.

From – To	**Signalling Message Size (bytes)**
CxB – SME	1700
SME – NME	2340
IPT – MTO	649
SME – Terminal	3105

Table 6 – Signalling Overhead evaluation for first session modification

Again, in these results, it is possible to conclude that the signalling messages exchanged between the IPT and the MTO are a small part of the total control information necessary to update the existing session. Upon the modification of the user context information, the CxB forwards it to the SME, which will match the updated context information with the existing content list and select the more appropriate codec. It this situation, a new codec is going to be selected. The list of supported codecs is then sent to the NME, where the file with the more appropriate bitrate considering the network situation is going to be selected. Then, NUM is going to be triggered to update the network paths for the new stream and forward it to the IPT, which will send to the MTO the necessary actions to update the existing enforcement. After successfully complete, the IPT returns with success to NUM, and the SME is informed that the session was updated. It then sends the necessary SIP signalling messages to the existing terminal, completing the session modification event.

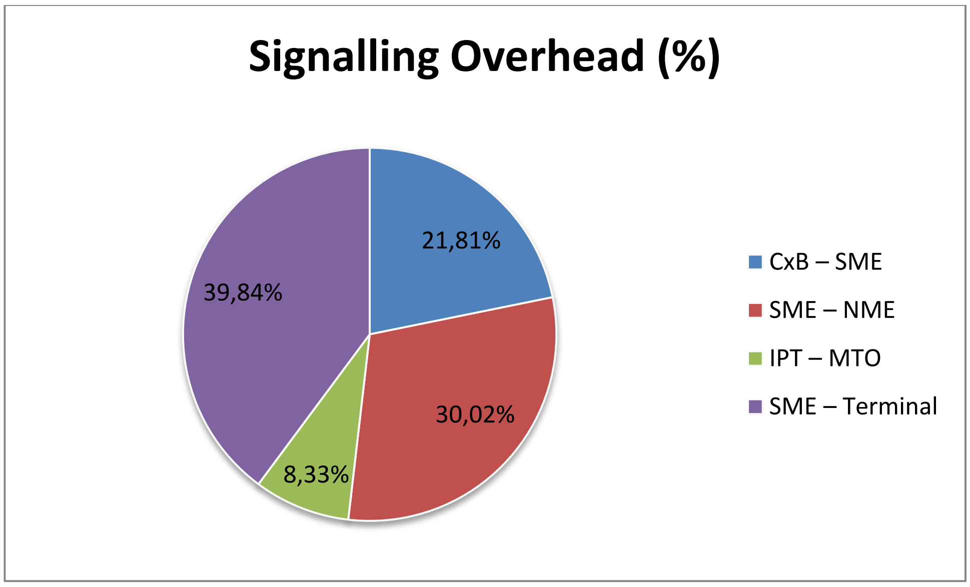


Figure 33 – Signalling overhead percentage for first session modification event

Considering that the value presented in Figure 33 contains both messages sent by the IPT and by the MTO, and that these messages are almost equal (as explained), the total signalling overhead introduced on the network by the IPT component is approximately 4.1%, which is a very low value comparing with the other components involved.

The third test is a new session modification event, but with a new user (connected through a multicast connection) in the user group. The results obtained for this event are presented in Table 7.

After the new user is placed in the user group by the GME and its multimedia content selected by the CSE, the SME is triggered to update the existing session. It is required to select the more appropriate codecs based on the new user context information. After these are selected, the NME is invoked to select the file with the best bitrate considering the network context information. NUM is then triggered to select the new path through which the multimedia contents should be forwarded to the new user. The new network setup is then sent to the IPT Controller. Since in this new test the enforcements of the network are going to be made by both MTO and IPT Node modules, the IPT Controller is required to exchange messages with both. After the new enforcements are successfully completed, the IPT Controller returns with success to NUM and the SME will have knowledge that the network enforcement was successfully updated. It then sends the necessary SIP signalling messages to the new terminal in order to enable the multimedia contents to be delivered to him.

From – To	Signalling Message Size (bytes)
CDE – SME	1157
CxB – SME	2357
SME – NME	2701
IPT – MTO	528
IPT Ctrl – IPT Node	404
SME – Terminal	3647

Table 7 – Signalling Overhead evaluation for second session modification

Again, as depicted in Figure 34, the messages exchanged between the IPT Controller and the MTO make up for approximately 5% (therefore 2.5% for the IPT Controller) and that the messages exchanges by both IPT internal components (IPT Controller and IPT Node) make up for another 3.7%, it can be concluded that the IPT was responsible for approximately 6.2% of the complete signalling overhead introduced in the network.

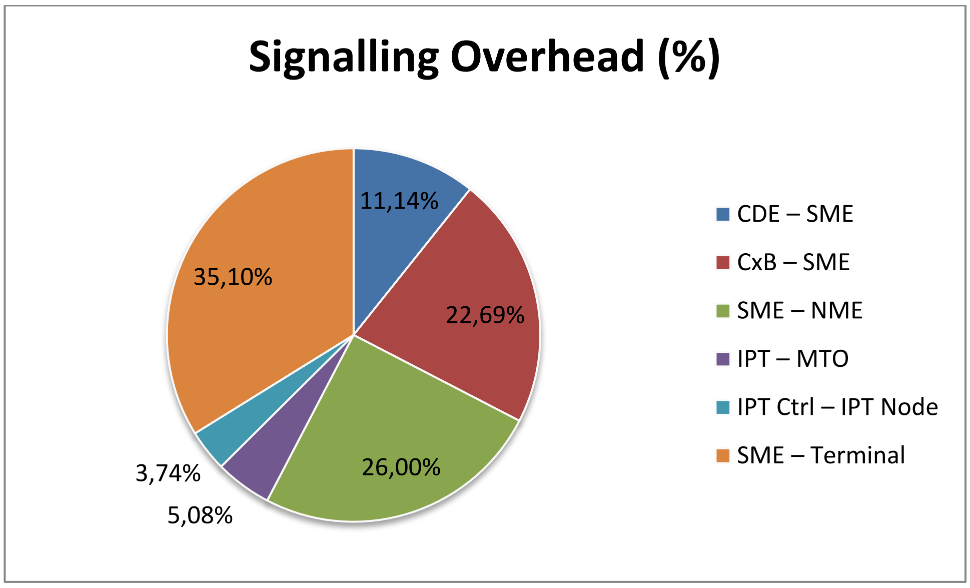


Figure 34 – Signalling overhead percentage for second session modification event

In the final test scenario, the existing session is going to be completely removed. Thus, the session management framework will request the release of the current enforcements in the network to the multiparty transport framework and send the necessary SIP signalling messages to terminate the multimedia sessions on the user's terminals.

This event starts with CDE triggering the SME component to terminate the existing session. However, unlike in the previous events, the SME is not required to interact with the CxB or the CtPD, and as such, it will only pass to the NME component the request to release the previously created enforcements in the network. Considering that in the previous tests studied the IPT was requested to perform various enforcements in the network through both the IPT Node and MTO modules, to release those reservations the IPT will have to exchange signalling messages with both modules, as presented in Table 8. After the release of the network enforcements is complete, the SME will send the necessary SIP messages to the streamer and to the user's terminals, informing them of the session's closure.

From - To	Signalling Message Size (bytes)
CDE - SME	1134
SME - NME	1248
IPT - MTO	155
IPT Ctrl - IPT Node	404
SME - Terminal	1002
SME - Streamer	756

Table 8 – Signalling Overhead evaluation for session remove

Observing the results depicted in Figure 35, it is possible to conclude that the signalling messages exchanged between the IPT and the MTO are almost irrelevant, only forming 3% of the total signalling communication necessary to terminate de session. The release of the multicast enforcements through the exchange of control information between the IPT Controller and the various IPT Node modules is more significant than the IPT-MTO communication. However, it is still inferior to the various interactions performed by the remaining architecture elements.

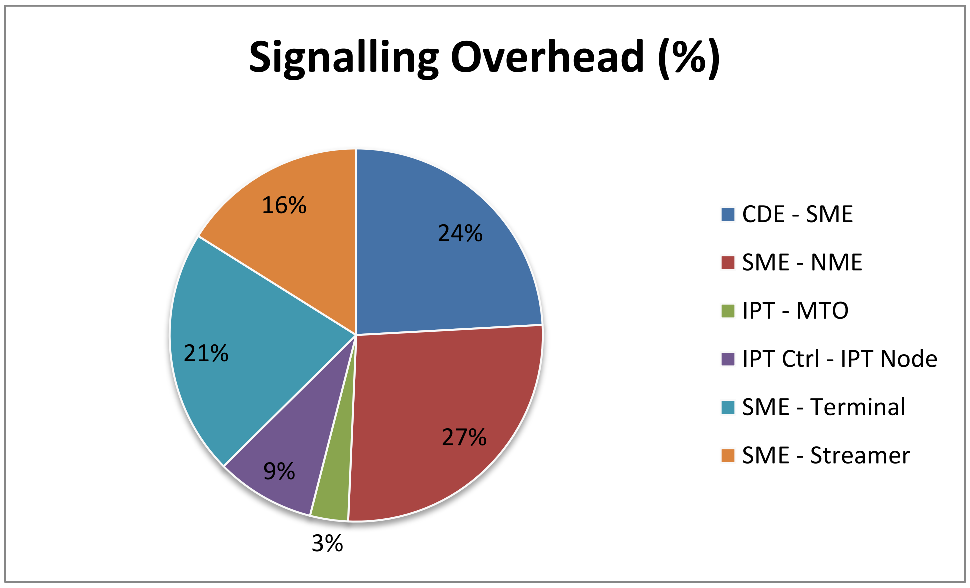


Figure 35 – Signalling overhead percentage for session remove event

Considering all the results obtained in this section, it is possible to observe that the IPT is not responsible for throttling the network performance, since the signalling overhead introduced in the network is considerably small in comparison with the remaining components in the specified architecture. Furthermore, considering that these control messages are only necessary upon specific triggers for session events, it can be concluded that the IPT does not have a noticeable impact on the network performance.

On the other hand, it is also possible to conclude that the signalling overhead introduced in the network by the IPT when specific session events are triggered is directly linked with the complexity of the NUM request. This is visible comparing the session creation and the first session modification events, where the network setup is the same: but while in the first request the IPT Controller had to create a completely new tree, in the second event the IPT Controller was only required to update the existing enforcements. Thus, the amount of signalling information exchanged in the first event was approximately twice the necessary for the second event. On the other hand, it is also possible to observe that the inclusion of the IPT Node module in the enforcement in the network also increases the necessary amount of signalling messages. However, even when this is the case, the total amount of signalling overhead introduced in the network is considerably smaller than the remaining components.

Considering the last test studied (session remove), it was visible a significant decrease in the amount of information exchanged between the IPT and the MTO. This is explained by the fact that, when removing previous enforcements through the MTO component, the IPT is required to send a *Remove_Mto_Tree* command. However, in this command, no information of the tree setup on the network is present, since the MTO only requires the ID number of the tree being removed. Thus, the actual control information exchanged for this event is significantly shorter. Additionally, considering that various nodes on the network had been previously enforced by the IPT Node modules, the IPT Controller was also required to release these reservations through the exchange of signalling information with the IP Node modules. However, even in this scenario the impact of the control information exchanged by the IPT is shorter than the ones of the remaining architecture elements, proving that the IPT does not deteriorates the overall architecture performance.

5.5. Processing Time Evaluation

Continuing the performance evaluation on the IPT, it is essential to estimate the processing time requirements for the IPT, the variables that affect this performance, and what is the impact of the IPT component on the overall architecture. To obtain these results, the main session events (session setup, session modify and session remove) are going to be subjected to a series of tests, allowing to estimate the processing time requirements and the respective confidence intervals at 95%. For each session event tested, the number of users being subjected to the event is also going to change from 1 to 3 users, thus allowing identifying the impact that the number of users being affected by the various events has on the required processing times. It is also important to note that on these first tests the only architecture elements being evaluated are the IPT, MTO and NME/NUM.

Starting the tests with the session setup event, the results obtained are show in Table 9 and Figure 36.

1 User

Component	Time (ms)	Time (%)
NME/NUM	15.2 ± 1.51	74,44%
IPT	0.44 ± 0.09	2,15%
MTO	4.78 ± 0.6	23,41%

3 Users

Component	Time (ms)	Time (%)
NME/NUM	17.4 ± 1.63	70,16%
IPT	0.6 ± 0.12	2,42%
MTO	6.8 ± 0.66	27,42%

Table 9 – Comparison of the processing times for each component in session setup

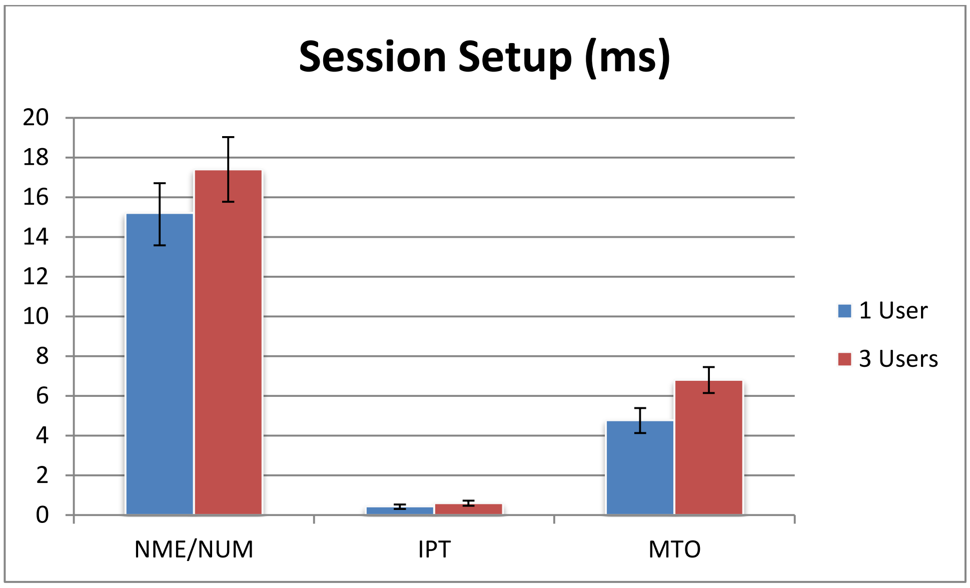


Figure 36 – Processing time for Session Setup event

Through the observation of these results, it is clear that the impact of the IPT component on the architecture performance is minimal, considering that it only requires approximately 2% of the total time. On the other hand, it is also visible that by increasing the number of users being subjected to the session events the processing times increase, although only slightly. Thus, it is expected that as the complexity of the network setups for the various session events increases, the processing times required by the various components to setup the multimedia session also increase. However, despite the complexity increase, the IPT impact on the overall performance is still minimal.

Continuing the performance evaluation, another session event was tested, the session modification. For this event, two different situations are going to be studied. The first corresponds to a scenario where only one user is subjected to modifications, while for the second scenario it is considered that three users are modified simultaneously. The results obtained for these tests are depicted in Table 10 and Figure 37.

1 User

Component	Time (ms)	Time (%)
NME/NUM	12.2 ± 0.66	81,44%
IPT	0.54 ± 0.04	3,60%
MTO	2.24 ± 0.23	14,95%

3 Users

Component	Time (ms)	Time (%)
NME/NUM	14.8 ± 1.16	77,27%
IPT	0.72 ± 0.1	3,76%
MTO	3,63 ± 0,47	18,97%

Table 10 – Comparison of the processing times for each component in session modify

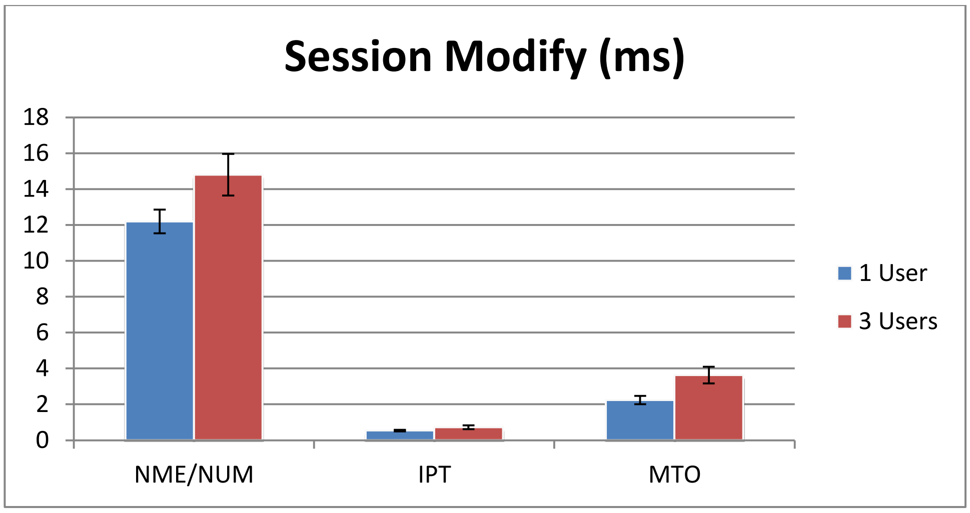


Figure 37 – Processing time for Session Modify event

Through a careful observation of the results presented, it can be concluded (in similarity to the results for the first test) that as the network setup complexity increases, the various components in the network are subjected to higher processing and signalling efforts and as such, the required processing times tend to increase. Thus, it is also expected that for the session modification event, an increase in the complexity for the setups requested leads to higher delays. However, it is also noticeable that in both situations the relevance of the IPT component in the total required time is almost insignificant, proving that the developed component does not introduce a significantly negative impact for the overall performance of the implemented architecture.

For the final event tested, the existing session is going to be removed, thus triggering a session remove event. Again, for this event both versions of one and three users are also going to be performed. The results obtained during the various tests for this session event are shown in Table 11 and Figure 38.

1 User

Component	Time (ms)	Time (%)
NME/NUM	14.4 ± 0.7	71,22%
IPT	0.3 ± 0.06	1,48%
MTO	5.52 ± 0.43	27,30%

3 Users

Component	Time (ms)	Time (%)
NME/NUM	15.6 ± 1.31	71,04%
IPT	0.32 ± 0.09	1,46%
MTO	6.04 ± 0.56	27,50%

Table 11 – Comparison of the processing times for each component in session remove

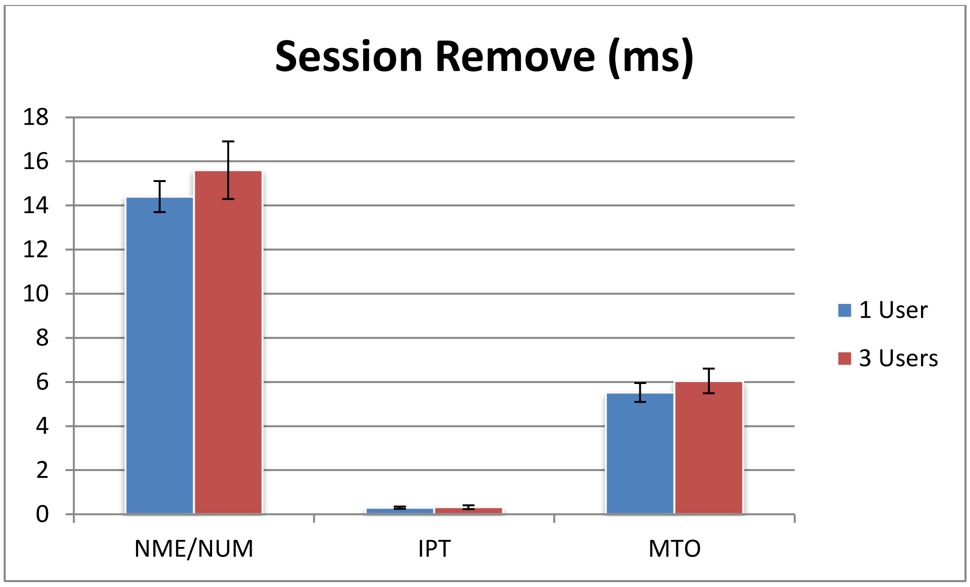


Figure 38 – Processing time for Session Remove event

Analysing these tests, it is possible to verify that the same conclusions as before can be applied for the NME/NUM and MTO. However, for the IPT it is visible that there are no significant differences in the required processing times when the network setup increases in complexity (1 user affected to 3 users affected). This can be explained by the IPT internal processing of this event. Considering that the control message exchanged with the MTO does not require any information relative to the various nodes and connections used to enable the delivery of the multimedia contents to the users, the messages created for these events are similar and do not require almost no possessing of the setups sent by NUM. Thus, the complexity of the requests does not have a significant impact on the IPT performance for the session remove event.

The various session events studied are not however, only influenced by the NME/NUM, IPT and MTO components. The session events require communications between other elements in the network before the path tree can be selected and enforced by these components. When the session event is triggered in the SME by the CDE (as explained in section 3.6), the SME has to request a list of the available formats and codecs from the CtPD and the group users context information from the CxB. A match between these lists is then done to select the more appropriate formats to be streamed to each user. Only then the NME is triggered to select the tree for this session and enforce it in the network through the IPT and MTO. After the enforcement has been completed in the network, the SUM is also required to manage the SIP signalling messages to the user's terminals, enabling the start of the reception of the requested multimedia contents. The time results observed for the SME component for a sample session event are presented in Table 12 and Figure 39.

Interface	**Time Spent (ms)**
CtPD	599
CxB	847
NME/IPT/MTO	23
SIP Signalling	38
Internal Processing	45
Total	*1552*

Table 12 – Time spent for the overall architecture for a sample session event

Observing these results, it is possible to conclude that the enforcement of the tree in the network represents a very small amount of time in comparison to tasks such as the communication between the SME and the CtPD and CxB. Furthermore, from all the interface communications performed by the SME, the one with the NME, IPT and MTO is the fastest.

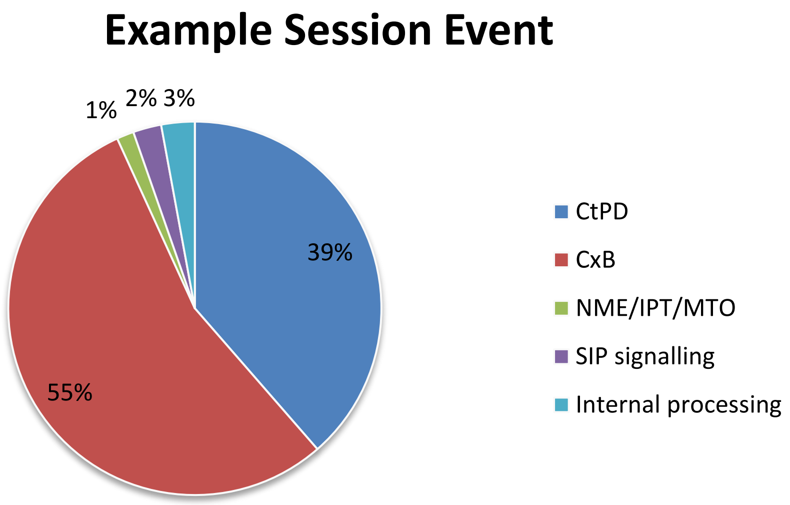


Figure 39 – Time percentage for the overall architecture in a sample session event

Thus, it is clear that the network setup selection and enforcement process is not responsible for relevant negative impact on the overall architecture performance (1% of the processing time required by all the interfaces on the SME). Furthermore considering the small percentage of time required by the IPT component for the enforcement of the requested trees on the network, it is clear that the developed module does not introduce significant delays in the establishment, modification and termination of the various sessions requested.

5.6. IPT Performance Conclusions

Considering the work developed in the IPT component and the various tests and integrations performed with it in the C-Cast project architecture, it is possible to confirm that the developed module is capable of performing the initial proposed objectives. Through its utilization, it is possible to enforce various tree setups requested by the SUM on the network. It is also possible to manage the recommended actions to be taken by the MTO and the distributed IPT Node modules. Finally, it is also capable of performing a dynamic allocation of the addresses for the various multicast groups on the network.

On the other hand, considering the results presented in section 5.4 and 5.5, it is also possible to conclude that the impact of the IPT in the network performance is minimal. Considering the signalling overhead results obtained, IPT does not significantly impact the network performance, since the necessary signalling messages exchanges make up for a small part of the total signalling necessary for each session event. On the other hand, this impact becomes event less noticeable when it is considered that these control messages are only exchanges upon specific session even triggers, and thus are rarely exchanged during the existence of a multimedia session. Considering possible variables that could affect the signalling overhead values, it was concluded that the network complexity and the necessity of enforcing trees with both components (MTO and IPT Node) was the major influence. Thus, it is expected that, as the network increases in scale and complexity, the necessary control messages exchanges increase. However, so the remaining components in the network would also be subjected to the same increase, and as such, the network performance would not be directly decreased because of the IPT component. For the processing times required by the IPT to enforce the various requests on the network, it is also clear that its impact will be minimal, and as such, not negatively influencing the total time required to establish, modify and terminate the requested sessions.

During the tests of the processing time used by the IPT, it was also clear that the number of users does not significantly impact the IPT performance.

5.7. Summary

This chapter presented several scenarios and situations used to evaluate the behaviour and impact of IPT module in the network.

It was verified that the implemented architecture is able to correctly enforce the session trees in the network, through the MTO and IPT Node modules. The trees enforced support multicast, unicast or hybrid connections, successfully enabling the delivery of media contents to groups of users independently of the underlying network. The support for dynamic sessions was also effectively implemented, considering that the IPT was able to adapt existing enforcements on the network to match the modifications to which the session was subjected. It was also proved in section 5.3 on the various test scenarios studied that the IPT was able to perform these modifications without incurring service disruptions to users who were not modified.

Despite the fact that the message overhead can vary, its impact in the network performance and in the delivery of the multimedia contents is minimal.

Finally, the processing time required by the IPT is considerably small in the multiparty transport framework, and almost insignificant considering to total time required by a session event. Thus, it is possible to confirm that the developed solution is able to implement its objectives and with an acceptable performance.

6. Conclusions and Future Work

The main purpose of this Dissertation was to develop and implement a new component, capable of enabling the delivery of media contents to groups of users, independently of the underlying network. Dynamic modifications to existing sessions should also be supported, being triggered upon updates to the network and/or connected users.

The proposed solutions were tested through several distinct scenarios deployed in a general testbed, proving that it was able to create, modify and remove successfully multiparty sessions. The implemented solution for the IPT guarantees that, through the hierarchical architecture proposed for the component, it is possible to have a central unit capable of processing several requests from the higher level components like the NME, and enforcing them on modules deployed throughout the network such as IPT Node and MTO. On the other hand, the module deployed in various nodes in the network (i.e. IPT Node) is also capable of creating, maintaining and removing specific paths through the network, enabling the improvement of quality of service on multicast transport through the enforcement of the best paths available accordingly to the network and user context information. The IPT solution implemented is also capable of successfully creating hybrid paths in the network, enabling multicast transport where it is supported for a better resource management, and unicast transport, where the network is not able to support multicast.

Through the tests realized and the results obtained, it is possible to demonstrate that the IPT component can be used to enable the delivery of multimedia contents to dynamic sessions without creating a negative impact on the network. It is also able to enforce and manage various multicast paths in the network while enforcing unicast paths through the MTO, creating the support for an independency of the underlying network capabilities. Support for the adaptation of the network to context changes was also developed, having verified that the IPT is able to modify existing tree setups with minimal service disruption to users who were not subjected to any modifications.

Considering the performance results obtained during the various test scenarios realised, it was verified that the component was able to complete its objectives without penalizing the network performance. Considering the control messages overhead on the network, it was verified that it had a considerably low value in comparison to various possible stream sizes. Furthermore, considering that these messages are only occasionally exchanged, they do not affect the delivery of the multimedia contents. On the other hand, evaluating the processing times required in comparison with the multiparty transport framework, it was verified that the IPT did not influence, since the other components required significantly more time. In comparison with the overall processing time required in a session event for the complete C-Cast architecture, it was verified that the IPT required less than 0.1% of the total processing time. Thus, it is possible to conclude that the performance results obtained are significantly positive.

However, the fact that the network was implemented based on a hierarchical structure hints at possible scalability problems. Since the central intelligent unit, IPT Controller, is responsible for the management of every IPT Node module deployed in the network, the presence of a significant number of these elements in the network can severely harm the normal operation of the IPT module, raising the processing times and message overhead in the network to unacceptable values. Hence, work is necessary to provide a more decentralized approach to the IPT component, distributing the knowledge of groups of IPT Nodes through more than one unit in the network, or by restructuring the existing architecture to a more distributed approach.

References

1. C-Cast project. [Online] http://www.ictccast.eu/. Acedido em 23 de Maio de 2010.

2. **Postel, J.** Transmission Control Protocol. *RFC 793.* 1981.

3. —. User Datagram Protocol. *RFC 768.* 1980.

4. **Adams, A.** Protocol Independent Multicast - Dense Mode (PIM-DM). *RFC 3973.* 2005.

5. **Fenner, B., et al.** Protocol Independent Multicast-Sparse Mode (PIM-SM). *IETF RFC 4601. 2006.*

6. **Waitzman, D.** Distance Vector Multicast Routing Protocol. *IETF RFC 1075.* 1988.

7. **Moy, J.** Multicast Extensions to OSPF. *RFC 1584, Proteon Inc. 1994.*

8. **Ballardie, A.** Core Based Trees (CBT version 2) Multicast Routing. *IETF RFC 2189.* 1997.

9. **Fenner, W.** Internet Group Management Protocol, Version 2. *IETF RFC 1112.* 1997.

10. **Cain, B., et al.** Internet Group Management Protocol, Version 3. *IETF RFC 2236. 2002.*

11. **Deering, S., Fenner, W., Haberman, B.** Multicast Listener Discovery (MLD) for IPv6 . *RFC 2710.* 1999.

12. **Lao, L. et al.** A Scalable Overlay Multicast Architecture for Large-Scale Applications. *IEEE Transactions on Parallel and Distributed Systems.* 2007, Vol. 18, pp. 449-459.

13. **Banerjee, S., Bhattacharjee, B., Kommareddy, C.** Scalable Application Layer Multicast. *In Proc. of ACM Sigcomm.* 2002.

14. **Yeo, C., Lee, B., Er, M.** Survey of application level multicast techniques. *Transactions of the Elsevier Computer Communications.* 2004. Vols. Vol. 27, I.15, pp. 1547-1568.

15. **Hosseini, M., Ahmed, D.,Shirmohammadi, S., Georganas, N.** A Survey of Application-Layer Multicast Protocols. *IEEE Communication Surveys.* 2007. Vol. Vol.9.

16. **Harrison, T., Williamson, C., Mackrell, W., Bunt, R.** Mobile multicast (mom) protocol: multicast support for mobile hosts. *MobiCom '97: Proceedings of the 3rd annual ACM/IEEE international conference on Mobile computing and networking.* 1997.

17. **Wang, Y., Chen, W.** Supporting ip multicast for mobile hosts. *Mob. Netw. Appl.* 2001. Vols. vol. 6, no. 1, pp. 57–66.

18. **Janneteau, C., et al.** Comparison of Three Approaches Towards Mobile Multicast. *IST Mobile Summit. 2003.*

19. **Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W.** An architecture for differentiated service. *IETF RFC 2475.* 1998.

20. **Wehrle, R. Bless and K.** Group communication in differentiated services networks. *Cluster Computing and the Grid.* 2001.

21. **Pereira, V., Mendes, P. , Monteiro, E.** Evaluation of an overlay for source-specific multicast in asymmetric routing environments. *IEEE GLOBECOM.* 2007.

22. **Wang, B., Hou, J.** Multicast routing and its QoS extension: problems, algorithms, and protocols. *Network, IEEE, vol. 14, no. 1, pp. 22–36.* vol. 14, no. 1, pp. 22–36, Feb 2000.

23. **Mendes, P.** OSMAR: Overlay for Source-specific Multicast in Asymmetric Routing environments. *NTT DoCoMo Euro-Labs.* 2004.

24. **Neto, A., et al.** A resource Reservation Protocol Supporting QoS-aware Multicast Trees for Next Generation Networks. *IEEE ISCC.* 2007.

25. **Bhattacharyya, S.** An Overview of Source-Specific Multicast (SSM). *IETF RFC 3569.* 2003.

26. **Jannotti, J., Gifford, D., Johnson, K., Kaashoek, M., O'Toole, J.** Overcast: reliable multicasting with on overlay network. *Proceedings of the 4th conference on Symposium on Operating System Design & Implementation.* 2000.

27. **Dey, A.** Understanding and using context. *Personal and Ubiquitous Computing.* vol. 5, pp.4–7, 2001.

28. **Thomas, R., Friend, D., DaSilva, L., McKenzie, A.** Cognitive Networks: Adaptation and Learning to Achieve End-toEnd Performance Objectives. *IEEE Communications Magazine.* 2006.

29. **Hancock, R., Karagiannis, G., Loughney, J., Van den Bosch, S.** Next Steps in Signaling (NSIS): Framework. *RFC 4080.* 2005.

30. **Buford, J.** Hybrid Overlay Multicast Framework. *Internet Draft. 2008.*

31. **Durham, D., Ed., Boyle, J., Cohen, R., Herzog, S., Rajan, R., Sastry, A.** The COPS (Common Open Policy Service) Protocol . *RFC 2748.* 2000.

32. **Rigney, C., Willens, S., Rubens, A., Simpson, W.** Remote Authentication Dial In User Service (RADIUS) . *RFC 2865.* 2000.

33. **Rigney, C.** RADIUS Accounting. *2866.* 2000.

34. **Calhoun, P., Loughney, J., Guttman, E., Zorn, G., Arkko, J.** Diameter Base Protocol. *IETF RFC 3588.* 2003.

35. **Janneteau, C., Simoes, J., Antoniou, J., Christophorou, C., Kellil, M., Klein, A., Neto, A., Pinto, F., Roux, P., Sargento, S., Schotten, H., Schneider, J.** Context-Aware Multiparty Networking. ICT Mobile and Wireless Communications Summit, 2009. *ICT-MobileSummit 2009 Conference Proceedings.* 2009.

36. **Neto, A. et. al.** Multiparty Session and Network Resource Control in the Context Casting (C-CAST) project. Future Multimedia Networking, 2009.

37. *Mrouted daemon.* [Online] http://publib.boulder.ibm.com/infocenter/aix/v6r1/index.jsp?topic=/com.ibm.aix.cmds/doc/aixcmds3/mrouted.htm. Acedido em 18 de Março de 2010.

38. **Janneteau, C., et al.** Specification of context detection and context-aware multiparty transport. *Deliverable D13.* 2009.