



Universidade do Minho

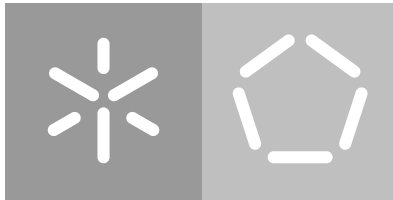
Escola de Engenharia

Departamento de Informática

André Filipe da Silva Sampaio

**Development of an Adaptable
Multicast Overlay Network**

December 2017



Universidade do Minho

Escola de Engenharia

Departamento de Informática

André Filipe da Silva Sampaio

**Development of an Adaptable
Multicast Overlay Network**

Masters dissertation

Masters Degree in Informatics Engineering

Dissertation supervised by

Professor Pedro Nuno Sousa

December 2017

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my Advisor, Professor Pedro Nuno Sousa, without whose guidance this work would not have been possible and whose help from start to finish has truly been exceptional.

I would also like to thank my colleagues and friends, particularly, Alexey, Ricardo and Vasco, who aided a lot during difficulties encountered along the way.

Last but not least, I would like to thank my family, specifically my parents and brother, who have helped me to reach this point, and who have supported and pushed me during the toughest times.

To all of you, my sincerest appreciation and thank you.

ABSTRACT

Multicast is a group communication paradigm created in order to reduce, as much as possible, the amount of data generated to the network. However, limited deployment of IP Multicast protocols has motivated an interest in alternative approaches which implement a similar process of Multicast at an application-level (using solely end-systems and not the routers). In this context, different methodologies are presented, entitled *Application-Layer Multicast* or *Overlay Multicast*, which may vary in the way they operate.

This dissertation's objective is to develop and experiment a prototype of an overlay multicast system. This system should be easily configurable and adaptable in order to assume different strategies when establishing the multicast distribution tree. It is also expected to explore and integrate collaborative mechanisms between the overlay network and the Internet Service Providers (ISP).

With the presented context, the first step to take is an investigation on the state of the art, where technologies relevant to this work will be presented. After this initial step, the developed system's architecture will be described, one which enables different ways of building and maintaining the multicast distribution tree. The envisioned system can operate independently, integrating mechanisms where the distribution tree relies solely on peer decisions, which will be firstly addressed. Then, this work will move on to collaborative mechanisms between the overlay's management (the central node) and the Internet Service Providers. Based on the proposed system architecture, several mechanisms are explored, not only focusing on alternative ways to build distribution trees, but also mechanisms allowing for some traffic engineering objectives involving the Internet Service Providers. Using the CORE network emulator, all the proposed mechanisms are tested, and results are analyzed to corroborate the system's correct operation.

RESUMO

O multicast é um paradigma de comunicação em grupo que tem como objetivo reduzir, tanto quanto possível, a quantidade de tráfego gerada para a rede. No entanto, a implantação limitada de protocolos IP Multicast tem motivado o interesse em abordagens alternativas que implementam processos de distribuição Multicast na camada aplicacional (ou seja, usando apenas os sistemas/aplicações finais e não os routers). Neste contexto, surgem as soluções denominadas por *Application-Layer Multicast* ou *Overlay Multicast*, podendo estas apresentar algumas variantes na sua operação.

Nesta dissertação, tem-se como objetivo o desenvolvimento e experimentação de um protótipo de um sistema de Overlay Multicast. Este sistema deverá ser capaz de ser facilmente (re)configurado para assumir diferentes estratégias no estabelecimento da árvore de distribuição Multicast, e integrar mecanismos de colaboração entre a rede Overlay e os Internet Service Providers.

No contexto apresentado, o primeiro passo consiste na investigação do estado da arte, onde tecnologias relevantes ao atual trabalho serão apresentadas. Após este passo inicial, a arquitectura do sistema será apresentada, uma arquitectura que considera diferentes maneiras de construir e manter a árvore de distribuição multicast. O sistema proposto pode operar de forma independente, contemplando mecanismos onde a árvore de distribuição depende apenas das decisões dos vários peers, sendo que estes serão os primeiros mecanismos a serem apresentados. De seguida, o sistema direcciona-se para mecanismos colaborativos entre a gestão da rede overlay e o ISP, de maneira a incluir conhecimento acerca da topologia da rede que nenhuma outra entidade seria capaz de providenciar. Com base na arquitectura do sistema proposto, vários mecanismos são explorados, não só mecanismos que se concentram em formas alternativas de construir a árvore de distribuição, mas também mecanismos que permitem cumprir os objetivos de engenharia de tráfico dos ISPs. Por fim, utilizando o emulador de redes CORE, todas as soluções serão testadas, e os seus resultados analisados por forma a validar a correta operação de todo o sistema.

CONTENTS

Acknowledgements	i
Abstract	ii
Resumo	iii
Table of Contents	iv
List of Figures	vii
List of Tables	x
List of Acronyms	xi
1 INTRODUCTION	1
1.1 Introduction and Motivation	1
1.2 Objectives	2
1.3 Main Contributions	3
1.4 Thesis Organization	3
2 STATE OF THE ART	5
2.1 IP Multicast	5
2.1.1 Multicast Fundamentals	5
2.1.2 IP Multicast	7
2.1.3 Multicast Groups	10
2.1.4 Multicast Routing Protocols	11
2.1.5 Protocol Independent Multicast (PIM)	13
2.2 Overlay Peer-to-Peer (P2P) Systems	15
2.2.1 Overlay P2P Concepts and Applications	15
2.2.2 Architecture	16
2.2.3 Consequences/Problems and Challenges	22
2.3 Application-Layer Multicast	24
2.3.1 Application-Layer Multicast Concepts	25
2.3.2 Illustrative ALM Works and Approaches	28
3 SYSTEM ARCHITECTURE AND DEVELOPED MECHANISMS	35
3.1 General Architecture	35
3.1.1 Central Node	36
3.1.2 Peer	40
3.1.3 ISP Collaborative Service	42
3.1.4 Extended Central Node	43
3.1.5 Extended Peer	44

3.2	Distribution Trees Construction	45
3.2.1	Minimum-Delay Approach	46
3.2.2	Minimum-Loss Approach	48
3.3	Collaborative Methods/Approaches	54
3.3.1	Link Protection	54
3.3.2	Link Minimization	57
3.3.3	ISP Forwarder Activation	61
3.4	Overlay in multiple Autonomous Systems	64
4	TESTING AND RESULTS ANALYSIS	67
4.1	Technologies and Testing Platform	67
4.1.1	Development Tools	67
4.1.2	Network Emulation	70
4.1.3	Web-Application	70
4.2	Minimum-Delay	71
4.2.1	Activating the Central Node	72
4.2.2	Activating the Collaborative Service	73
4.2.3	Activating the Sender (Node N8)	73
4.2.4	Activating a Receiver (Node N7)	74
4.2.5	Activating other Receivers	75
4.2.6	Deactivating Peer N25 (10.0.22.20)	78
4.3	Minimum-Loss	79
4.3.1	Activating the Scenario	80
4.3.2	Activating Receiver N7	81
4.3.3	Activating Receiver N28	82
4.3.4	Activating Receiver N30	83
4.3.5	Activating Receiver N29	85
4.4	Protect-Link	86
4.4.1	Passive Protection	86
4.4.2	Active Protection	92
4.5	Link Minimization	97
4.5.1	Activating the Scenario	98
4.5.2	Link Minimization Procedures	98
4.6	ISP Forwarder Activation	102
4.6.1	Activating the Scenario	105
4.6.2	Forwarder Activation Step 1 - Link Minimization Procedures	105
4.6.3	Forwarder Activation Step 2 - Activating Forwarders	106
4.7	Multiple Autonomous Systems	108
4.7.1	Activating the Scenario	110

4.7.2	Link Minimization Procedures and new Distribution Tree	113
5	CONCLUSIONS	115
5.1	Developed Work	115
5.2	Future Work	116
6	BIBLIOGRAPHY	119
A	SECONDARY TREE REPRESENTATION	123
A.1	Minimum-Delay Approach	123
A.2	Passive Link Protection	123

LIST OF FIGURES

Figure 2.1	Unicast vs Multicast	6
Figure 2.2	Multicast Groups Example - Part One, Extrapolated from [2]	9
Figure 2.3	Multicast Groups Example - Part Two, Extrapolated from [2]	10
Figure 2.4	API - Interface for Peers, Source: [19]	18
Figure 2.5	Flooding-Based search schema, Source: [6]	20
Figure 2.6	Random Walk search schema, Source: [6]	21
Figure 2.7	BitTorrent Architecture, Source: [19]	22
Figure 2.8	IP Multicast delivery, adapted from Source(s): [4, 43, 45].	26
Figure 2.9	Application-Layer Multicast delivery, adapted from Source(s): [4, 43, 45].	27
Figure 2.10	TOMA architecture, Source: [12]	28
Figure 2.11	OMNI architecture, Source: [34]	30
Figure 2.12	Scattercast architecture, Source: [36]	32
Figure 3.1	Conceptual Architecture	36
Figure 3.2	Conceptual Central Node view	37
Figure 3.3	Distribution Tree example, Central Node's Graph Representation	39
Figure 3.4	Conceptual Peer View	40
Figure 3.5	Conceptual Central Node, Extended	43
Figure 3.6	Conceptual Peer, Extended	45
Figure 3.7	Activity Diagram - Creating a Multicast Session	46
Figure 3.8	Joining Peer Decision Example	49
Figure 3.9	Creating a minimum-loss multicast session	50
Figure 3.10	Minimum Loss Answer Accumulator	52
Figure 3.11	Joining a minimum-loss multicast session (Simplified)	53
Figure 3.12	Passive Link Protection	56
Figure 3.13	Complete Graph Representation In Parity Maps	58
Figure 3.14	Complete Graph Matrix Representation	59
Figure 3.15	Forwarder Placement	63
Figure 3.16	Best Match Determination Logic, N Forwarders	63
Figure 3.17	ISP Forwarder Activation Summarized Process	64
Figure 3.18	Multiple Autonomous Systems - Expanded Architecture	65
Figure 4.1	Graph Data Structures	69
Figure 4.2	Testing Network Topology	71

Figure 4.3	Activating the Central Node	73
Figure 4.4	Creating a multicast session	73
Figure 4.5	Activating Node 7 - Part 1	74
Figure 4.6	Activating Node 7 - Part 2	74
Figure 4.7	Activating Node 7 - Part 3	75
Figure 4.8	Activating Node 7 - Part 4	75
Figure 4.9	Activating Node 7 - Part 5	75
Figure 4.10	Receivers Activation, Distribution Tree Representation	76
Figure 4.11	Overlay Distribution Representation	76
Figure 4.12	Path between N8 and N30	77
Figure 4.13	Deactivating Peer N25 Logs	78
Figure 4.14	Deactivating Node N25 - Tree Representation 1	79
Figure 4.15	Updated Network Topology	80
Figure 4.16	Activating Scenario Result	81
Figure 4.17	Logging - Arrival of Peer N7.	82
Figure 4.18	Logging - Arrival of Peer N28.	83
Figure 4.19	Minimum-Loss Tests: Multicast Tree Stage 2.	83
Figure 4.20	Logging - Arrival of Peer N30.	84
Figure 4.21	Minimum-Loss Tests: Multicast Tree Stage 3.	85
Figure 4.22	Logging - Arrival of Peer N29.	85
Figure 4.23	Minimum-Loss Tests: Multicast Tree Stage 4.	86
Figure 4.24	Activating the Central Node	87
Figure 4.25	Activating the collaborative service - Part 1	87
Figure 4.26	Activating the collaborative service - Part 2	88
Figure 4.27	Logs regarding the activation of the sessions' sender	88
Figure 4.28	Logs regarding the activation of peer N7	88
Figure 4.29	Multicast Tree Stage 1	89
Figure 4.30	Path from peer N25 to peers N7 and N8	89
Figure 4.31	Multicast Tree stage 2 - Representation	90
Figure 4.32	Passive Protection: Overlay Distribution Tree	90
Figure 4.33	Central Node's reply to peer N26	90
Figure 4.34	Multicast Tree Stage 4	91
Figure 4.35	Passive Protection: Final Overlay Distribution Tree	91
Figure 4.36	Tree before protection (Stage 7)	93
Figure 4.37	Active Protection: Overlay Distribution Tree	93
Figure 4.38	Path between peers N29 and N30	94
Figure 4.39	Collaborative Service's request to protect link N5-N18	94
Figure 4.40	Central Node Logs Receiving the collaborative service's request	95

Figure 4.41	Central node identification of problematic peers	95
Figure 4.42	Peer N29 logging the removal of peer N30 from downstream list	95
Figure 4.43	Peer N30 is requested to establish new connection	95
Figure 4.44	Peer N30, paths to possible connections	96
Figure 4.45	Final multicast tree	96
Figure 4.46	Active Protection: Overlay Distribution Tree	97
Figure 4.47	Scenario activation resulting Overlay Distribution Tree	98
Figure 4.48	Scenario activation result	99
Figure 4.49	Parity Map Creation (summarized)	100
Figure 4.50	Complete Graph Matrix	100
Figure 4.51	PRIM output for MST	101
Figure 4.52	Logging - Central Node logs peer Notifications and acknowledgements.	101
Figure 4.53	Final Multicast Tree.	102
Figure 4.54	Final multicast tree with demonstrated paths.	103
Figure 4.55	Adapted Network Topology	103
Figure 4.56	Scenario activation result	105
Figure 4.57	Distribution Tree Representation	106
Figure 4.58	Link Minimization result	106
Figure 4.59	Distribution Tree Representation 2	107
Figure 4.60	Forwarder Tests Report	108
Figure 4.61	Forwarder Activation result	108
Figure 4.62	Multiple Autonomous Systems Network Topology	109
Figure 4.63	Scenario activation result	111
Figure 4.64	External Graph Matrix	113
Figure 4.65	Final Distribution Tree	113
Figure 4.66	Final distribution tree - Paths	114
Figure A.1	Receivers Activation Representation 2	123
Figure A.2	Deactivating Node N25 - Tree Representation 2	123
Figure A.3	Multicast Tree Stage 4 -Representation 2	124

LIST OF TABLES

Table 2.1	Packet count log unicast	7
Table 2.2	Packet count log multicast	7
Table 2.3	Most common Structured and Unstructured P2P overlays, summarized from source [9]	17
Table 3.1	Entities and interactions of Figure 3.1	36
Table 4.1	Link introduced data delay	72
Table 4.2	Packet Loss assumed in specific links	79

LIST OF ACRONYMS

- ALM Application-Layer Multicast.
- ALTO Application-Layer Traffic Optimization.
- AS Autonomous System.
- ASM Any Source Multicast.
- DHT Distributed Hash Table.
- DM Dense Mode.
- DVMRP Distance Vector Multicast Routing Protocol.
- HTML HyperText Markup Language.
- IANA Internet Assigned Numbers Authority.
- IETF Internet Engineering Task Force.
- IGMP Internet Group Membership Protocol.
- IP Internet Protocol.
- MSN Multicast Service Node.
- MSON Multicast Service Overlay Network.
- OLAMP Overlay Aggregated Multicast Protocol.
- OMNI Overlay Multicast Network Infrastructure.
- P₂P Peer-to-Peer.
- PIM Protocol Independent Multicast.
- QOS Quality of Service.
- RFP Reverse Path Forwarding.

RIP Routing Information Protocol.

RP Rendezvous Point.

SCX Scattercast Proxy.

SM Sparse Mode.

SSM Source Specific Multicast.

TOMA Two-Tier Overlay Multicast Architecture.

TTL Time-To-Live.

INTRODUCTION

1.1 INTRODUCTION AND MOTIVATION

In the recent past, with the evolution the network infrastructure has seen, both in its capacity to deal with more and more users as well as the capability to deliver data much faster than ever before, the number of devices and applications that are now generating traffic to the network has increased exponentially. Considering this growth, a necessity arises for solutions and technologies which are able to cope with the ever increasing amount of data that is introduced in the network at any given moment, particularly with applications related to group communication, such solutions will have to be both efficient and scalable. In this context, the IP Multicast [1, 2] technology emerges as one of those solutions. In fact, it allows for a better exploitation of the network's resources, taking into consideration that a given data flow may be intended for various destinations, and IP Multicast allows for that same data to be replicated only in the branching nodes within the network [3].

Even though the benefits of implementing multicast at the network level (IP multicast) would be immense, the difficulties associated with such deployment [4] have lead to the development of new paradigms, such as Application-Layer multicast [10]. Among these complications emerge the complexity of the protocols associated with IP Multicast as well as the lack of scalability necessary for this technology to be applied to the Internet full extent [12] where a considerable amount of multicast users/groups will have to be supported. Also, economical and security reasons have been behind the non-deployment of multicast by Internet Service Providers.

On the other hand, Application-Layer multicast solutions are characterized by their rather easy implementation, considering its up to the application level developer to implement the system. Moreover, this kind of implementation does not result in a direct (economical) cost to the Service Providers given the fact that the management of multicast groups and data replication happens by software and not on routers, reducing the necessity for routers with increased computational capacity. However, these systems lack the network state knowledge, which leads to a scenario where the multicast implementation is not always as efficient as possible.

All the facts stated above have led to a stagnation of IP Multicast development and to an approximation to the overlay network alternative. In this context, this thesis aims to develop and implement an overlay multicast system with an easily reconfigurable distribution tree, in order to assume different contexts of usability, and which enables interaction between the overlay system and the ISP, who knows the current state of the network. This cooperation between the multicast system and the ISP provides the tools for a better management of the network's resources, enabling ISPs to apply different policies that better suit the network state and the kind of group communication that is being performed. This hybrid approach's objective is to allow for a better traffic management by the ISP, while also allowing for an increased knowledge of the application level, leading to a performance improvement with regards to a more resilient and efficient overlay infrastructure.

1.2 OBJECTIVES

This thesis main goal is to develop an Overlay Multicast network which is adaptable to different usability contexts. This overlay network is to be easily reconfigurable in order to be able to assume different behaviours in the establishment and maintenance of the multicast distribution tree taking into account distinct approaches (e.g. minimum delay, packet loss, computation cost, etc).

In addition to the previously mentioned objective, the prototype should include collaboration mechanisms between the overlay network and the Internet Service Providers in order to allow the ISPs to be active participants in the definition of the overlay network, providing the tools for the ISP to ensure its best interests and objectives. Finally, in a latter stage, the aim is to devise a solution that provides the overlay network with the capacity to operate in scenarios that involve various autonomous systems.

Taking into consideration the previously mentioned objectives, a list of particular objectives is presented:

1. Investigation of the *state of the art* of the involved research areas.
2. Definition of the various components and mechanisms to develop, as well as their operating rules, mainly with regards to the construction and maintenance of the multicast tree.
3. Implementation of the overlay network prototype. This implementation should represent all created entities and be able to establish the mentioned communication capacity between the various components of the system.
4. Test of the developed prototype as well as a comprehensive analysis of the obtained results. These tests are to, as much as possible, resemble a real scenario operation.

1.3 MAIN CONTRIBUTIONS

This work studied, designed, developed and tested the creation of a flexible and adaptable overlay multicast network. This network, with different possible configurations in order to maximize different objectives, presents an interesting proposal to the Internet Service Providers as the platform also allows the inclusion of the ISP in the management of the overlay. As the overlay network assumes a collaborative perspective, the ISP can cooperate with the system to better manage the overlay's traffic, and so, allowing the ISP to better implement its traffic engineering policies. This cooperative approach, which many other overlay architectures do not consider, improves the underlying topology's resilience and allows the overlay to enjoy the ISPs good will. This is possible as they operate together, not only to improve the overlay's performance but, more importantly even, to improve the ISP's capacity of peacefully adapting traffic flow into the overlay. Otherwise the ISP will have to take more drastic measures to protect the network topology.

The testing stage has shown proof that the developed architecture and methodologies operate properly, showing the operational feasibility of the system. This testing has shown peers forming an overlay network independently of the collaborative service whenever no intervention was required. Furthermore, when requests arrived on the central node, by the ISP, the performed tests show the central node manipulating the distribution tree in order to accommodate the ISP's operating commands.

1.4 THESIS ORGANIZATION

The current thesis will be organized in six chapters, whose description follows:

- **Introduction and Motivation:** This chapter provided a contextualization for the work to develop, having characterized the general concept of the idea as well as the problems a prototype like the one presented could solve and, most of all, the advantages in performance, scalability and security a system like this would provide.
- **State of the art:** This chapter will provide the theoretical premises with regards to the current state of the different technologies that will be used, in order to allow for a better understanding and definition of the various components and mechanisms that will have to be conceived and that will result in the proposed solution. In this chapter, a broad look into multicast, and particularly multicast at the network level, will be provided, as well as a more in-depth analysis of overlay peer-to-peer systems and application-layer multicast.
- **Architecture and Developed System:** Here, a detailed look is given on the developed architecture and distribution tree construction methodologies. First, the sys-

tem's general architecture is presented and described, namely the main entities and components, as well as the way they operate on a logical level. After this description, different approaches on the construction and maintenance of the multicast distribution tree are presented, where both independent and collaborative approaches are shown and explained.

- **Testing and Analysis:** The testing stage shows the technologies used in the development of the present work's prototype, as well as a variety of test cases/scenarios. Initially, the development tools are presented as well as the testing platform, the CORE network emulator. After, a test case is presented for each of the operating methodologies, showing the obtained results, as well as a critical analysis as to the system's response to events, which are shown step-by-step.
- **Conclusions:** This chapter presents the conclusions extracted from this work. The developed work is summarized before presenting what could be the future work for the presented system, namely improvements to be made that would definitely be an addition to the system's characteristics and extend its range of operability.
- **Bibliography**

STATE OF THE ART

This chapter will provide an analysis of the fundamental concepts regarding the technologies important to this thesis. First and foremost, an explanation on the way IP multicast operates in order to provide an idea of why this way of implementing multicast has not seen a great evolution or a great interest by the Service Providers, Section 2.1. Having presented the fundamental concept of multicast and IP Multicast, the focus will shift to peer-to-peer overlay systems, in order to introduce a different paradigm to the way of distributing data, Section 2.2. Finally, an investigation of application-layer multicast systems is presented, in order to gain the necessary sensibility to understand the current ways of forming multicast distribution trees so as to determine proper paths for data flows, Section 2.3.

2.1 IP MULTICAST

This section will detail IP Multicast, the way multicast distribution trees are formed as well as the way they are maintained [2]. However, before specifying IP Multicast, an explanation will be provided on the Multicast paradigm itself, mainly its principles and objectives.

2.1.1 *Multicast Fundamentals*

Multicast is a technology developed with a very important purpose in mind, which is to reduce, as much as possible, the amount of data generated into the network. In a world where group communication is increasingly more important (be it via video-conferencing, stock data, gaming and so on) [3], the amount of data that applications are introducing in the network is immense. Typically, should a device, for some reason, need to send any data to another device, a unicast connection is established, meaning that packets go from one source to one destination. This makes perfect sense, given the fact that such source is trying to reach a specific destination. However, considering the case where a given source is trying to send the same exact information to two (or more) destinations, what should happen? In a typical scenario, the sender creates as many datagrams as there are destinations, a unicast connection is established between the sender and each of the receivers, and all datagrams

are sent to and through the network. Multicast removes this necessity on the sender by introducing a new strategy in data-delivery, the sender now simply creates one datagram with one destination group and sends it to the network, and this single datagram is, then, replicated only in branching nodes and delivered to all recipients in that group. Figure 2.1 illustrates this exact paradigm difference.

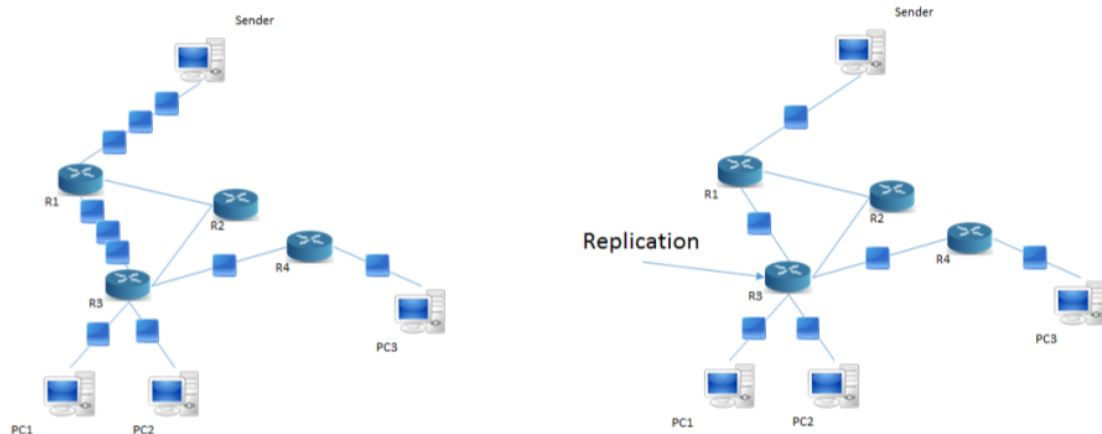


Figure 2.1.: Unicast vs Multicast

Analyzing Figure 2.1, on the left side, a given sender is attempting to send the same information to PCs 1, 2 and 3. As unicast connections are in place, the sender creates three copies of the same packet, one for each destination, and sends them into the network, through router R1. This router, then, forwards the data to router R3 who delivers one copy to receivers PC1 and PC2 each. Router R3 also forwards one copy of the data to router R4, with final delivery to PC3 taking place. Alternatively, on the right side of Figure 2.1, where multicast is being used, the employed logic is quite different as the sender sends only one copy of the data into the network, with the multicast group's address. So, router R1 forwards the only copy of data it received to router R3. This router, knows receivers PC1 and PC2 are directly connected to it, replicates the packet twice, one for each receiver, and forwards the packet to router R4, who, once more, makes delivery to PC3, performing no replication.

Tables 2.1 and 2.2, represent the number of datagram packets that are in the network at any given time. Considering a scenario where receivers PC1, PC2 and PC3 are not in a multicast session/group, the sender needs to create three new datagram packets and these datagrams need to travel, separately, through the network until they reach their destination, which does not happen in a case where receivers PC1, PC2 and PC3 are in a multicast group and the sender simply needs to create one single datagram and send it to the group. When comparing Tables 2.1 and 2.2 an immediate conclusion is established, which is that multicast

Sender	Receiver	Number of Packets
Sender	Router R1	3
Router R1	Router R3	3
Router R3	PC1	1
Router R3	PC2	1
Router R3	Router R4	1
Router R4	PC3	1

Table 2.1.: Packet count log unicast

Sender	Receiver	Number of Packets
Sender	Router R1	1
Router R1	Router R3	1
Router R3	PC1	1
Router R3	PC2	1
Router R3	Router R4	1
Router R4	PC3	1

Table 2.2.: Packet count log multicast

reduces the amount of data in the network structure and also reduces the amount of work the sender needs to perform, dividing the work throughout the network infrastructure itself.

2.1.2 IP Multicast

Having seen a very broad description regarding multicast's modus operandi, it's important to take a more detailed approach, particularly on a network level, IP Multicast. As stated in Section 2.1.1, generally normal IP packets are sent, via unicast connection, from a single source to a single recipient, where said datagram packets are delivered to the destination according to a predetermined forwarding table. Taking into consideration that the number of established connections equals the number of receivers, and also that the number of (equal) datagram packets the source will have to create also equals the number of recipients, it is clear that the efficiency of this process is not the best, given the current growth on the (ever increasing) number of applications that stream audio and video [14]. Also, in order to be able to create a datagram for each receiver, the source would need to maintain an updated list of all receivers, which is unfeasible, taking into account the processing cost of such a task. More importantly even, on a network level, what this means is that a number of copies of the same data would be introduced into the network and would flow through the same links, which means a huge increase of bandwidth usage and also costs due to

the fact that the network itself would have to be able to deal with such a huge data flow increase.

Thus far, it has been established multicast provides the source with a way to deliver data to all destinations sending only a single copy of data which routers then forward, constantly duplicating said data only in branching nodes, meaning, where paths to different destinations vary. However, how do these routers know where to send this data to? Well, multicast groups identify a set of recipients which are interested in data being streamed by a particular source [2]. Moreover, these multicast groups have an (attributed) IP address and the source simply sends data to this IP address which is then forwarded to all members of the multicast group.

Taking into consideration the concept of multicast groups, it is important to understand how these procedures are actually put in place, mainly with regards to the job each entity has to perform. Clearly there are different approaches each entity takes when interacting with these groups:

- The source sends only one datagram packet into the network, whose destination address is the IP address of the actual multicast group, not having to consider how this data is actually delivered to all destinations;
- Users that wish to receive the data being streamed, need to join the multicast group the sender is addressing. This is achieved through a join request to a multicast router in the user's local interface, using an Internet Group Membership Protocol (IGMP).
- Multicast routers are in constant communication between themselves, and implement Multicast Routing Protocols (such as PIM-SM [15] and PIM-DM [16]) so as to guarantee not only that all hosts that have joined each group receive their wanted information but also that no bandwidth is wasted sending data to multicast routers that do not contain any receivers in the designated multicast group.

It is clear that all this information regarding multicast groups, mainly which multicast routers have group members appended to themselves, needs to somehow be maintained in an updated manner. So, multicast routing protocols constantly calculate a multicast distribution tree in order to ensure that traffic is not sent to unnecessary networks (networks that have no members in the multicast group) and to minimize the number of copies of the same data introduced into the same network link.

Let's take the small example in Figure 2.2 into consideration, assuming the Routing Information Protocol (RIP) is the protocol being used in the computation of the respective forwarding tables.

In this scenario, PC₁, PC₂, PC₄, PC₇ and PC₈ are members of the multicast group the source is sending to. And so, a few conclusions can be drawn, such as:

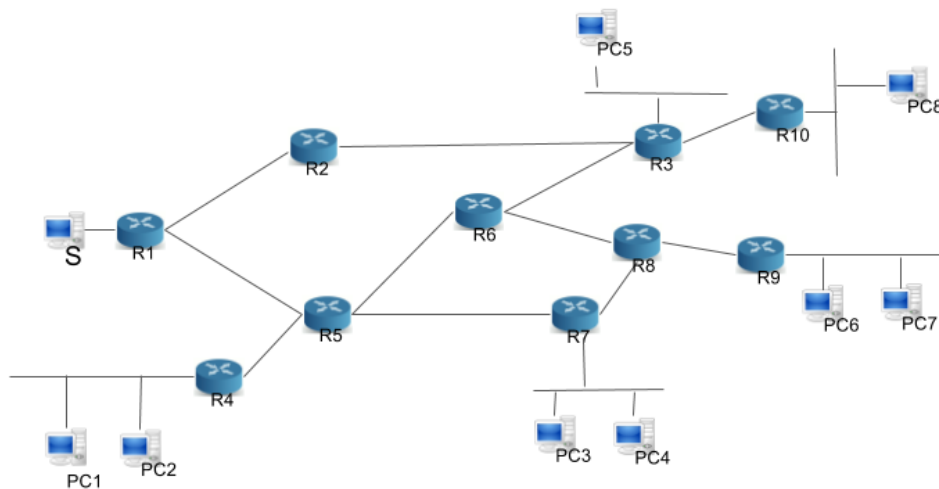


Figure 2.2.: Multicast Groups Example - Part One, Extrapolated from [2]

- The source sends only one datagram to router R1;
- Router R1 duplicates the datagram (it is a branching node!) and sends one copy to router R2 and another to router R5;
- Router R2 forwards the data to router R3 which then forwards it to router R10;
- Router R10 delivers the datagram to PC8, and stops;
- Having received the datagram from router R1, router R5 duplicates it and delivers one copy to router R4 and another to router R7;
- Router R4 duplicates the datagram once more, and delivers a copy to PC1 and another to PC2;
- Router R7 performs another copy of the same datagram and delivers one copy to router R8 and another directly to PC4;
- Router R8 forwards the data to router R9, not performing any copies;
- Router R9 delivers the data to PC7 directly.

Figure 2.3 illustrates the described process, highlighting copies performed by the multicast routers, i.e., highlighting the branching nodes.

However, what would happen should PC3 now join the same multicast group? Would the number of datagrams in the network change? No. Simply, router R7 would just perform yet another copy upon receiving data from the multicast group in question, and deliver that copy directly to PC3, which is in the same local network as PC4.

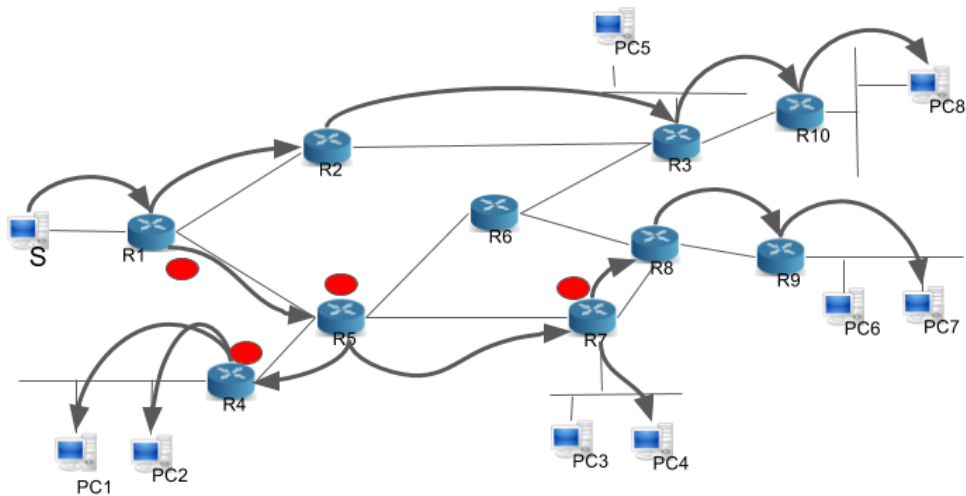


Figure 2.3.: Multicast Groups Example - Part Two, Extrapolated from [2]

2.1.3 Multicast Groups

It has been stated that the multicast source creates only one datagram packet and addresses it to the multicast group it is streaming to, and data is then carried out to all corresponding members. However, no description has yet been given as to the way these multicast groups are created or to the way receivers discover them.

Well, before a given receiver starts receiving data from said source (or list of sources) it must first determine the address of said multicast group. The Internet Assigned Numbers Authority (IANA) [17] is the entity responsible for assigning multicast group addresses to pre-established protocols and services. Then, network administrators offer other addresses in order for hosts to be able to request and use them, mainly at an application level.

Having established how multicast groups are assigned for a given time to a host (in this case, a sender), it is now important to mention that there are different types of hosts, receivers for instance. So, when joining a multicast group, hosts can elect to receive data sent to the group from multiple sources (ASM - Any Source Multicast) or from a specific source (SSM - Source Specific Multicast).

Any Source Multicast

Taking into consideration a scenario where a given host wishes to join a multicast group and receive data from all sources within that group, this host needs only to specify the IP address of the multicast group in question ($*,G$), and begins receiving data from any host acting as a source. However, even though the process to enter the multicast group and start receiving data is much simplified, it comes with a few drawbacks:

- Security - The ASM model is more susceptible to denial-of-service attacks, due to the fact that sources are not controlled and may keep flooding the network with an excessive amount of data;
- Interdomain Multicast Routing - In this scenario, and remembering that there can be a multitude of sources, it may become harder to deal with the multicast tree formation and maintenance. This comes as a result from the fact that the mechanism for source discovery will necessarily have to be much more complex.

The ASM model is quite used for applications where the number of sources is not controlled, predictable, or keeps changing. Applications such as video-conferencing for instance.

Source Specific Multicast

The Source Specific Multicast scenario presents quite a number of differences in the way it operates when compared with ASM. When entering a multicast group, a given host must not only specify the IP address of the multicast group itself but also the IP address of the source it wishes to receive data from (S,G), which, obviously, adds another layer of complexity (and necessary knowledge) on the part of the joining member. Now, this added complexity also comes with a number of advantages [18]:

- Elimination of cross delivery of traffic when more than one source simultaneously use the same source specific destination address;
- Address allocation becomes much simpler since the multicast group address is local to the source, meaning no global allocation mechanism is required;

It becomes clear SSM is more appropriate for scenarios where it is predicted that the number of sources will be quite small, and the sources are unchanging. Consider the example of audio and video broadcasting where, despite the number of receivers, the sources typically remain the same.

2.1.4 Multicast Routing Protocols

Multicast routers need to exchange information between themselves, mainly regarding their directly connected neighbours, in order to be able to join and leave multicast trees, taking into consideration the multicast groups that hosts appended to them are connected to. These information exchanges are performed using a multicast routing protocol, of which there are a few. However, the most used protocols in this area are Protocol Independent

Multicast Sparse Mode (PIM-SM), Protocol Independent Multicast Dense Mode (PIM-DM) and Distance Vector Multicast Routing Protocol (DVMRP).

The mentioned communication between multicast routers, is used in order not only to allow multicast routers to join and leave multicast groups, but also in the creation and maintenance of multicast trees so that, once a multicast tree has been constructed, data is forwarded down the multicast tree, replicated when the path to the different recipients diverges.

So, knowing multicast trees are constructed using multicast routing protocols, it is important to understand the paradigm differences that make these various protocols diverge. The multicast tree is constructed according to three main factors: *i)* The routing protocol used (opt-in or opt-out); *ii)* Their tree construction and maintenance mechanism (source-based or shared-tree); *iii)* Their upstream router determination.

OPT-In vs OPT-Out Protocols

Protocol Independent Multicast Sparse Mode (which will be further detailed) is an opt-in protocol. Opt-in (or sparse) protocols are used in scenarios where the receivers are considered to be sparsely distributed throughout the network. This means that, if routers were to be receive data they did not ask for, many subnets would be receiving a lot of multicast packets [2]. Well, in an opt-in protocol, routers are inserted in the multicast tree solely when they indicate interest in a determined data flow, otherwise, are kept out of the multicast tree. It is important to state that, in this approach, routers send a join message to the upstream router only when one host, that is appended to said router, asks to join the multicast group.

Whereas with opt-in protocols, routers have to express their will to be included in the multicast tree, in the case with opt-out protocols, such as Protocol Independent Multicast Dense Mode and Distance Vector Multicast Routing Protocol, the default scenario is that every router in the network is interested in receiving the multicast data being streamed, and so, this data is forwarded to all routers. The logic being that, if a router does not have any host interested in said data, it will simply inform the upstream router of its wish to remove itself from the multicast tree, which is done via prune request message.

Source-Based vs Shared Tree Protocols

Multicast trees can be constructed in various manners, depending on the chosen multicast routing protocol. There are two types of trees used by the main multicast routing protocols, they can be source based or shared based.

In the case of source-based tree protocols, an independent multicast tree is built for each source (each sender) in the group, leading to a multitude of different trees. Each one of these trees finds its root in the router to which such source is connected to. PIM-DM and

DVMRP are source-based tree protocols, which means, routers wishing to join a particular multicast group must specify both the IP Address of the group and the source they are interested in, a request that is made to the upstream router.

Shared tree based protocols, present a more complex system when it comes to tree generation. This is due to the fact that, instead of creating a new multicast tree for each source, only one multicast tree is generated, possibly more complex, and all sources in the multicast group use the same multicast tree to forward their data. This single tree is rooted at an elected node (the *Rendezvous Point (RP)* in PIM). However, in a scenario where there is only one root and multiple sources are enabled to stream data into the multicast group, in order to ensure every host receives the desired data, it is fundamental that all data be distributed at the source of the tree. This is performed by sending the data to the root of the tree at which point it is forwarded through the appropriate paths in the network

Finally, it is important to highlight that shared tree protocols are more appropriate than source-based protocols when there are more potential sources in the multicast group, however, they require a more inefficient system due to the fact that, before being forwarded to all hosts, datagrams must be delivered to the root of the multicast tree.

Determining the Upstream Router

It has been stated that, when joining a multicast group, routers must determine their upstream multicast router. This requires reverse path forwarding (RFP) look-ups, which can happen in order to determine the address of either the source of data or the root of a shared tree, depending on the model being used.

Multicast routers use the same interface for their join and prune messages (or any control packet) as well as content, i.e., actual data being sent. To perform their reverse path forwarding look-up operations, most protocols use their own mechanisms to exchange the necessary routing information (such as DVMRP) while others, such as PIM, use a Multicast Routing Information Protocol populated by a third-party source.

2.1.5 *Protocol Independent Multicast (PIM)*

Protocol Independent Multicast is a collection of multicast routing protocols, each optimized for a different environment. There are two main protocols, previously mentioned in this document, PIM Sparse Mode and PIM Dense Mode. All these protocols share a common control message format, messages which are sent either to the multicast group including the PIM routers, or as unicast datagrams to specific locations.

In this section a brief description of the two main versions of the protocol will be presented, focusing particularly on the advantages and disadvantages of each protocol.

Protocol Independent Multicast Sparse Mode

PIM Sparse Mode is an opt-in multicast routing protocol, once more, meaning that routers must explicitly notify their upstream neighbours of their interest in a particular group and source. This notification is performed via PIM Join and Prune messages to join and leave the multicast routing tree. These PIM join messages have to keep being re-transmitted as PIM-SM is a soft-state protocol, which means, all state is timed-out after a predetermined time interval.

By default, PIM-SM uses shared trees, with the trees for each group rooted at a router called the Rendezvous Point (RP). As stated, data is sent from a source to the RP in PIM control messages using unicast connections. If necessary, PIM-SM can also fallback to source-based trees to avoid the encapsulation process required in order to send the message/data to the RP, among other reasons.

The advantages of PIM Sparse Mode far outweigh the disadvantages, which is why PIM-SM is considered to be one of the best solutions to the multicast paradigm, its advantages are:

- Like the name suggests, PIM is protocol-independent, meaning it can operate regardless of the unicast protocol that is implemented in the network;
- It scales well across bigger networks;
- Sparse Mode allows for information to be only contained at routers belonging to the multicast tree;
- It supports both SSM and ASM.

Even though the number of disadvantages is less considerable, it is important to understand that in shared trees, the process of encapsulation and decapsulation that is performed between the source and the Rendezvous-Point can become quite inefficient, and it may be required a number of times.

Protocol Independent Multicast Dense Mode

PIM Dense Mode (PIM-DM) is an opt-out multicast routing protocol. It is quite less common than PIM-SM due to the fact that it is most efficient in smaller domains, not scaling well when the network starts expanding. The fact that PIM-DM is an opt-out protocol, means that any router that joins the multicast group, is immediately added to the multicast distribution tree and, therefore, immediately starts receiving data.

Once more, PIM-DM, unlike PIM-SM, uses source-based trees in order to calculate its most efficient data forwarding scheme. There is, however, a particular rule about this protocol, which is that it does not have a mechanism in which routers explicitly join the

multicast tree, instead, they are presumed as branches of the multicast tree and required to send Prune messages in order to be removed from it.

To summarize the advantages of PIM-DM, remembering this is a smaller-scale protocol:

- It is a very efficient protocol when receivers are densely distributed in the network;
- It is protocol-independent;
- It supports SSM and ASM;
- It does not use Rendezvous points which makes it lighter and easier to implement.

2.2 OVERLAY PEER-TO-PEER (P2P) SYSTEMS

In the previous section, 2.1.2, regarding IP Multicast, even though, performance wise, the implementation of multicast at the network level provides a performance other solutions can not match, it is clear that this implementation is also quite limited. This has to do with various factors, such as the high amount of processing power IP Multicast would require from the multicast routers, for instance. Also, in a scenario contemplating any source multicast, denial of service attacks are clearly extremely hard to avoid, so security is also a limitation to the deployment.

In this context, other solutions came to light, further away from the network and closer to the application layer, most of which based on overlay p2p paradigms.

This section will present an overview of some peer-to-peer paradigm fundamentals, focusing on the solutions this approach provides. Then, different architectures will be contemplated, such as Structured and Unstructured P2P systems and finally an observation regarding the most used systems in a structured architecture as well as a fundamentals look on the basic search mechanism for unstructured P2P systems.

2.2.1 *Overlay P2P Concepts and Applications*

Peer-to-Peer overlay systems are distributed computer architectures, built on top of an existing network [6, 20], designed with multiple objectives in mind, such as sharing data, resources, etc, by direct exchange. This approach disregards a centralized server in the network coordination. In this context, without any hierarchical organization, the interconnected computers, named Peers, form self organizing overlay networks, operating on top of the Internet Protocol itself.

The concept of having independent peers forming and managing their own network comes with a number of advantages. The cooperative manner with which peers operate and share their resources, allows for a level of scalability and growth that, otherwise, would

be very hard to achieve. Peers perform a wide variety of tasks in a system like this, needing to act both as clients and servers at the same time while also participating in the process of searching and delivering content that they neither possess nor want, but it may be the case that, in the overlay, the path from source to destination includes those particular peers. Another very important characteristic of this kind of system has to do with its high fault tolerance and low vulnerability, namely its capacity to deal with peer failures. This is possible due to the fact that every peer operates independently, and, at most, what can happen is that certain content may disappear with the failing peer (in case only it should have the desired content). To this point, an attack on the network, would have to be of massive scale, and reach a large percentage of the overlay network, at the same time. Also, as peers operate independently, and this structure is peer-based, it makes for an inherently easier deployment when compared with a system that controls data flow on a network level.

Certain characteristics may vary in the construction of the overlay network structure, in order to optimize (namely with distinct routing and maintenance algorithms) it to achieve different levels of performance considering different goals, such as:

- Data-Sharing [23, 24]: Data storage and retrieval are one of the highest contributors to peer to peer systems development. Peers containing information spread it throughout querying peers (a logic that will be explained further in this document);
- Bandwidth-Sharing and Telephony[7, 25]: Very similar to data-sharing, but optimized for efficient streaming of real-time data throughout the network as the ability to shuffle information via different paths in the network becomes of major importance so as not to flood a specific path, and in order to take advantage of less occupied routes;
- CPU-Sharing [26, 27]: Peers can combine their machines, mainly during *down-time*, and perform data mining with applications being developed mainly for scientific research.

2.2.2 Architecture

As peer to peer systems may serve different purposes by having different applications and as they can be optimized to perform better under specific circumstances, it makes sense to think that this kind of optimization is to be reflected in the overlay's structure. As stated in the previous section, peers forming the overlay network operate independently but also, each peer has its own set of responsibilities, requiring them to perform a wide variety of tasks within the system, namely acting as both clients and servers to the overlay structure, performing their own direct connections to other peers (directly connected peers are called neighbours) and participating in search and content transmission efforts.

Structured	Unstructured
Chord	Freenet
CAN	Gnutella
Pastry	FastTrack
Tapestry	BitTorrent
Kademlia	UMM

Table 2.3.: Most common Structured and Unstructured P2P overlays, summarized from source [9]

Taking into consideration the different operating scenarios an overlay network can present, as well as peer obligations within the overlay, P2P systems are usually classified into two categories: structured and unstructured. Now, conceptually, each peer maintains its own collection of information/data to share, and, upon the arrival of a request for data, it begins transmitting it. Moreover, to perform a request, peers make query requests, making them the *querying peer*. When a peer receives a query, it checks its own data collection to see if it may act as the source for the desired information, and if so, begins transmitting, otherwise, forwards the request through its neighbours until either a peer with the desired data is found or a predetermined number of max hops is achieved and the search process is stopped. However, this process is not equivalent in all systems, which operate differently, with the most common structured and unstructured peer to peer overlays being presented in Table 2.3.

Structured P2P Overlay Networks

Structured p2p overlay networks offer a scenario where the network topology is controlled, and content is spread throughout strategic locations in order to maximize the search queries performance [19], instead of data simply being placed randomly throughout the network itself. The way this placement is performed, is according to an hash function that couples data-keys with data-objects. With this relationship established, a distributed hash table (DHT) is used to route key-based queries efficiently to the peers that strategically hold the data to retrieve. This concept ensures access to the desired data within a bounded number of hops from querying peer to the peer that holds the data.

Even though the presented ideology behind DHT-based systems is the same, which is to maximize network efficiency when it comes to querying peers, different DHT-based systems present different organization schemes for data objects and key space routing strategies. Despite this fact, DHT-based systems are expected to ensure the location of data objects happens, on average, with a $O(\log(N))$ efficiency, considering a number of hops metric and N as the number of peers in the actual overlay.

Despite the fact that, theoretically, the concept of DHT-based systems presents itself as an optimal solution with regards to querying within the overlay peers, it may also produce

an outcome where search performance is highly affected, taking into account performance searches may lead to choking points as all searches are directed to the same entity, adding inherent latency to the process. Figure 2.4 provides a visual representation of the API Interface such systems use.

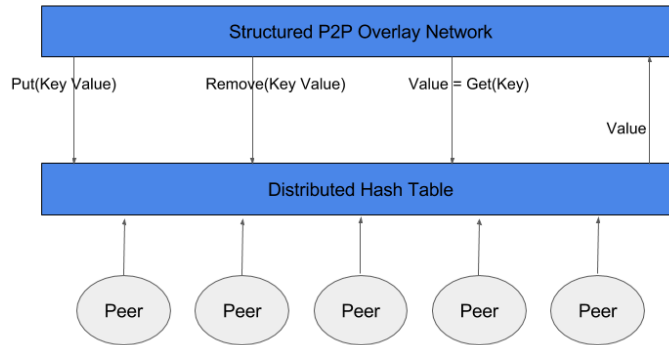


Figure 2.4.: API - Interface for Peers, Source: [19]

ROUTING Routing in structured overlay networks is widely different from unstructured overlays. The way queries conceptually operate has been very briefly mentioned in Section 2.2.2. However, it becomes clear that, as more and more peers join the system, the number of messages that need be exchanged increases exponentially, which will lead the physical network to reach its capacity, making the system non-scalable.

To deal with the presented problem, routing in structured overlays takes a different, more elegant approach, despite requiring more logistical work. Peers, on arrival, are attributed a unique identifier, which allows for each exchanged message to have a destination identifier. Each peer, sends/forwards this message solely to one neighbour in order not to flood the network. The selected peer is the next hop neighbour so as to minimize the number of hops from source to destination. This greedy approach to routing allows for a better management of the physical network’s resources, as the number of exchanged messages is quite smaller.

THE NAPSTER MODEL Napster [19, 21, 22] idealized an architecture for a peer-to-peer file-sharing system based on a centralized file search mechanism. File distribution changed from a paradigm where a central service was a single source of data to a scenario where popular content can be distributed by peers who have the searched content. A p2p file-sharing system such as this, presents a self-scaling solution as more and more peers join the system, taking into account that a peer who retrieves some desired content then becomes a source of the same content, increasing the download capacity, meaning, theoretically, that

a higher number of peers intent on getting the same content, make for a higher download speed as different hosts get their files and become instant sources.

This self-scaling objective is achieved via a centralized search facility to which every peer contributes. As a part of the overlay structure, every peer provides a list of content it possesses and, therefore, is able to distribute. With a list from each peer, this search system is able to establish a connection between content and network placement and, therefore, provide the querying peer with the identity of the peer it will download the requested content from. Obviously, this centralized search also creates a single point of failure for the whole system as an attack on the entity responsible for handling this search makes determining the source of data impossible for querying peers. Also, an unexpected and immediate growth in the number of querying peers would result in a system overload.

PASTRY Pastry [9] implements a DHT, similar to Chord [8, 28], to perform its network routing duties. Keys and nodeIds are stored sparsely throughout a number of selected hosts, as it is a self-organizing and fully decentralized system. Each pastry node maintains a set of neighbour nodes in its nodeId space, in order to locate the destination in each routing hop as well as to protect the network and provide it with a fault-tolerance system [5] by maintaining replicas of data items so peer failures may be dealt with.

Unstructured P2P Overlay Networks

An unstructured *p2p* system is built upon a different paradigm when compared to a structured scenario. In this case, there are no constraints on the relations between different nodes, and, as such, the overlay graph does not have a particular structure, meaning, content may not (and most likely is not) be distributed in the most efficient manner as peers connect to other peers randomly and there is no relation between the placement of the data objects with the network topology [20]. This leads to an approach where peers have limited (or even none) information on data objects stored by other peers, which results in a more complex search mechanism, as follows.

UNSTRUCTURED SEARCH SCHEMES Taking into consideration that, in an unstructured scenario, peer connections happen randomly and no central table of contents exists connecting peer identities with content they hold, searching in unstructured networks, for instance, is performed by flooding the network with a given query for content. Each peer visited will evaluate the query locally on its own content [19] and, should it have the desired data, reply as such. Otherwise, when a peer does not contain the content to retrieve, it forwards the query through its direct neighbours, meaning, the peers it has an established connection with.

Unstructured p2p networks are easier to implement when compared with structured scenarios and result in a much lower running cost as there is no need for a highly capable computational entity to reply to all search queries. However, as the number of participating peers grows, the number of exchanged messages grows with it. Even though this leads to a more resilient system as an attack would have to be equal to the scale of the network (if one peer fails, only its content is lost, all other peers remain accessible), this type of search, to the network itself, is quite expensive.

Flooding-Based Search In an unstructured p2p system, where content is not indexed in any way, the most basic search process is to flood [20] the network with a given query until the peer that contains the desired content is found. Flooding is performed by each peer forwarding the query to all neighbours. This makes up for a search system that can not grow with the network as the number of messages eventually gets too excessive. To stop the infinite growth on the number of messages exchanged, as would be the case in a pure flooding approach, a time-to-live is attached to the message in order to specify a maximum number of hops a message can perform. Figure 2.5 provides a visual representation on the Flooding-based search model.

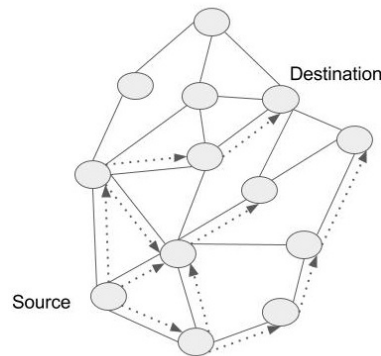


Figure 2.5.: Flooding-Based search schema, Source: [6]

Routing Indices based Search Routing Indices are not a necessarily different approach to a flood based search, they simply represent a more direct path to the peer that contains the searched content. Every peer, besides providing content it contains, also keeps a (rather small) index that stores the direction the query message should take so as to discover the intended data. This does not change the overlay's structuring system as content is still kept in each peer and is not organized in any particular way, simply nodes are contemplated with a higher knowledge of data placement within the network.

Random Walk A Random walk through the network graph comes as an optimization to the search process [29]. The previously regarded search schemes for unstructured overlay networks, contemplated a search system that had every peer, upon receiving a request/-query, check its own data structure and, should it not have the desired content, forward that request to all its neighbouring peers, most likely resulting in many duplicated requests. This comes at a cost to the physical network itself, as it is flooded with excessive requests, and the search process would only stop in case content was found or every message's Time-To-Live (TTL) was achieved.

The random walk search process has a slightly different approach. To contain the number of requests sent into the network, each peer forwards the received request, if necessary, to (only) one of its neighbours, chosen randomly. With this scenario, while it is assured that the network will not be flooded as only one copy of the request is in the network at any given time, the delay in delivery becomes unacceptable as there are no guarantees that the request is traveling in the right direction along the network. Figure 2.6 provides a visual representation on the Random Walk search model.

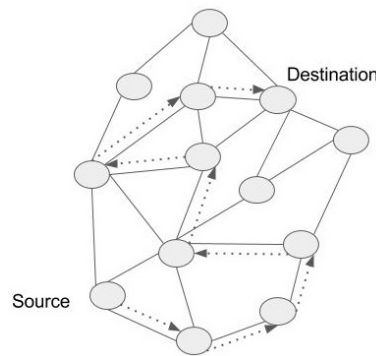


Figure 2.6.: Random Walk search schema, Source: [6]

BITTORRENT BitTorrent is widely known and one of the most popular P2P file sharing systems [9], having millions of users active at any given time. While heavily centralized, as it contemplates a central node/manager or tracker, it is considered to form an unstructured overlay network as peer connections do not follow any specific rules.

As far as the system's architecture, represented in Figure 2.7, it contemplates a central node, or tracker, which will manage the way downloads are performed. A *.torrent* file is downloaded, containing information of the data to download like file name, locations, etc [19]. The tracker keeps a record of all peers that have the requested file and the downloading peers connect themselves to those. As files are split in parts, any downloading peer can be receiving data from many sources and can also act as the source for the data it has already downloaded.

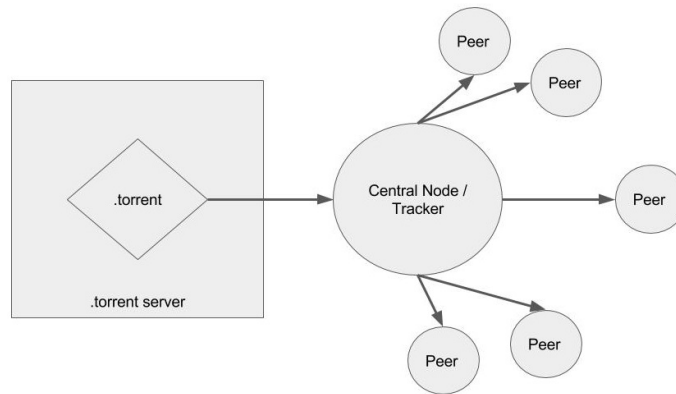


Figure 2.7.: BitTorrent Architecture, Source: [19]

A very interesting concept is the fact that bitTorrent, in order to maintain a level of fairness in the system, implements an upload-download direct connection, as peers with high upload speed will also probably have an high download speed and the download speed will be lesser should the peer have low upload capacity. So, *the more you share the more you get*, which can also be unfair in some ways, so, to limit peer download rate distinctions, bitTorrent also implements a choking algorithm to, temporarily limit peers from uploading, in order to maintain a consistent download rate throughout the whole network.

2.2.3 Consequences/Problems and Challenges

Having presented the reasons that lead to the existence of P2P systems and why they rose to such a scale, it is important to understand the additional consequences of their use.

In the remainder of this segment, a small overview will be given on the resilience of overlay networks to attacks and also on measurements to stop those attacks. Overlay maintenance and P2P-ISP collaboration issues will also be addressed.

Attacks and Vulnerabilities

The fact that, in p2p systems, peers operate independently, as mentioned, acts a huge amplifier for scalability but also provides a high level of resilience to the system, as the effect of a failing node does not represent a big problem to the network (only in the case where the failing peer was the sole carrier of specific data, in which case it may be a problem to other peers but not to the network) and an attack would have to be in large scale, affecting as many peers as possible in order for the network to be seriously harmed.

Despite the network's high level of resilience, attack security is still very much an issue and source of vulnerability as most current overlays are not secure, and malicious nodes can

easily insert themselves into the network and obstruct correct data delivery throughout the overlay and, in some cases, actually provide other peers with wrong (unwanted) messages. Typically good nodes can also be compromised and begin misrouting, corrupting or even dropping messages. When a good node is compromised, another major concern has to do with the information it is/was sourcing, as it may be deleted from the system or replaced with another under the same name, thus beginning a flow of corrupted data.

To the previous problems, security proposals come mainly in the form of secure routing, table maintenance and message forwarding [8]. Approaches that many systems are yet to adapt and that need careful attention when electing a system's structure, architecture and operating procedures.

Overlay Maintenance

Overlay networks, due to their characteristics and usability, are inherently highly dynamic and changing systems. High churn (churn regards the entrances and departures from the overlay), for instance, is a defining trait of such systems. Moreover, peers do not usually notify other peers of their departure, normally leaving the system abruptly or even dropping their connection and establishing a new one. To this point, the directly connected peers often are not aware of their neighbour's departure for quite some time and so, keeping an updated routing-table becomes a harder task.

To the presented point, overlay maintenance comes with significant importance in maintaining this type of flexible and adaptable network, as it is ever-changing. So, two main approaches have been deployed [6]:

- **Proactive Maintenance:** Peers periodically run updates to their own routing tables with regards to the topology in place. As an example, Chord periodically runs what is called a stabilization protocol to ensure peers are linked to other in increasing distance;
- **Reactive Maintenance:** In this kind of approach, peers react immediately to other peers failures or departures. Missing entries (due to the leaving peer) are replaced with new ones by sending connection signals to other established peers. Peer detection happens by *i) Probing:* Peers continuously run a ping-response protocol with each neighbour; *ii) Usage:* When a given peer messages another but no response arrives within a specif time-interval, that peer is considered to have failed.

P2P-ISP Collaboration

P2P overlay networks were developed with an intent to be used as a content distribution system where peers represent both the source and clients of data. This means a given peer is available to distribute a particular set of data upon request. However, P2P overlay systems

are formed on top of the Internet Routing architecture [38] with little (if any) knowledge of the underlying network structure.

Even though P2P systems may represent a source of revenue to the Internet Service Providers as users upgrade their internet service plan, they are also an enormous traffic engineering problem due to the fact that the amount of traffic inserted into the network is very high and, a lot of times, the routing scheme is implemented independently by the overlay structure and does not consider Internet Routing or the network topology [10]. So peers can not be used strategically throughout the network in order to reduce, as much as possible, not only the number of packets in the network but also, the length of the path datagrams take from source to destination.

A scenario to be considered is where peers A and B may be in the same autonomous system (or the same ISP) and peer C may be in another, however, as far as the overlay structure is concerned, peer A may be closer to peer C and may chose to exchange data with peer C. This means traffic would be crossing network bounds unnecessarily, increasing the operating cost to both Service Providers involved. Even worse, most bottlenecks in the Internet are assumed to be either in the access network or in links between ISPs [38].

In this context, it is important to provide both the overlay system and the Internet Service Provider with a way to collaborate in order to, at the same time, improve the performance of the p2p system and also allow the ISP to perform traffic engineering as it sees fit. IETF ALTO [46], created in 2008, is a IETF Working Group with this strategy in mind. The concept is to provide the application level with information that only the Internet Service Provider is capable of knowing, by way of a topology graph, and so, enabling the overlay structure to perform a mode sensible routing system, increasing the system's performance and decreasing the operating cost to the Internet Service Provider. With such a system, the ISP can use this mechanism to better manage the overlay traffic in the network by directing data along the fastest path possible, by avoiding possibly congested links, or by way of other strategy adopted by the ISP. Some additional research work examples, also involving collaborative efforts, between p2p application level and ISPs can be also found in [30, 31, 32].

2.3 APPLICATION-LAYER MULTICAST

In the sequence of sections 2.1 and 2.2, a few conclusions can be made on the different usability of both technologies.

In Section 2.1, the benefits of implementing multicast technologies become evident. Data delivery becomes much more efficient, both at the network level and for the source of content. By replicating data packets only when paths to different users diverge (branching nodes within the network) it is ensured the source no longer needs to keep a live-state on

all peers that wish to receive the data it is streaming. Also, this means the source needs only produce one data packet instead of producing one datagram per receiver, introducing the idea of one copy of data per network link. Despite all these advantages, IP Multicast has not evolved to a point where it is widely used due to the fact that these advantages come with a substantial operating cost as multicast routers require a high computational performance as well as extended state information of the multicast groups.

In Section 2.2, a different paradigm is presented as hosts act as both servers and clients. This means that there is not necessarily one single source within the whole system but as many sources as owners of content. So, different clients can retrieve the same content from different sources. The way the search for this content is performed is based on two separate architectures, structured and unstructured, which have been previously analyzed.

The present Section will provide a general perspective on an additional paradigm that takes inputs from both the previously presented systems. This paradigm, entitled Application Layer Multicast, tries to merge the two approaches, establishing itself in the frontier of the two areas. First, the ALM concept will be addressed in its operating logic and, then, some illustrative approaches will be presented and described.

2.3.1 *Application-Layer Multicast Concepts*

Application-Layer Multicast (ALM), in some ways, comes as a result of the sparse implementation of IP Multicast, an approach much closer to the network. As seen, this sparse deployment has to do with many factors, both technological and non-technological, namely the requirements that come with an IP Multicast implementation such as the necessity of maintaining a per group state [4], the overhead addition that comes with, for example, having each router maintain an entry corresponding to each multicast group, and finally, the non-existence of a pricing model that makes IP multicast feasible at this point.

Despite the mentioned IP Multicast deployment issues, the growth of network applications that require an ever-rising amount of messages exchanged (such as audio/video streaming [35] in the one-to-many department or video conferencing and multiplayer gaming where many-to-many communication is required, [44]) has been unquestionable. This has to do with its immense efficiency as a transmission mechanism capable of limiting the weight to the network. In this context, different alternatives for group communication services have been proposed, with Application-Layer Multicast providing an approach with a much easier level of deployment as it does not depend on a full knowledge of the network.

The Application-Layer Multicast concept relies on building overlay multicast trees (which can be built with the intention of maximizing performance to different contexts) among session participants, while connecting these participants (named peers) via unicast connections.

These peers are, then, responsible for forwarding data to other peers also participating in each session [39, 42].

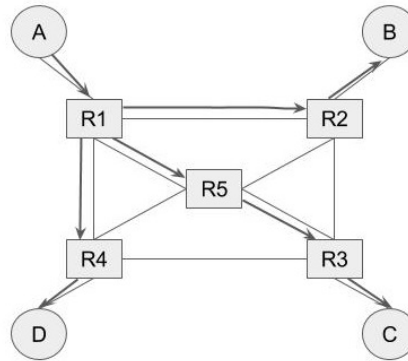


Figure 2.8.: IP Multicast delivery, adapted from Source(s): [4, 43, 45].

Figures 2.8 and 2.9 represent the two different approaches. Whereas in IP Multicast any duplication of packets happens in branching nodes within the network (and is performed by multicast capable routers themselves), in ALM, peers form direct unicast connections between themselves and transmit data directly to each other, with some peers acting as forwarders, i.e., as multicasting performers, meaning data replication is done in specific peers which are tasked with distributing data to a specific sub-set of peers in the session. Figure 2.8, shows Peer A as the source to the session, with router R1 performing three copies of the datagram packet (as it represents a branching node) and sending it to the respective receivers B, C and D. Figure 2.9 shows peer A (the source) creating two separate datagram packets, sending one copy to peers B and D each, and peer B is tasked with performing a copy of the received data and sending (forwarding) it to peer C. The latter Figure shows the created overlay, where the upstream to peers B and D is peer A, and the upstream to peer C is peer B.

Taking into consideration the concept that ALM is based on the overlay network peers themselves form, it is important to understand that the way data is distributed throughout the overlay can happen in different manners, meaning it can also be designed in such a way that it maximizes specific uses such as, for instance, minimum-delay [13], maximizing the network's throughput [40, 41], among other alternatives. The easy deployment and configuration to different contexts, as well as the fact that the overlay scales very well as the low overhead of dealing with arriving and departing peers leads to an increase in usage of this type of architecture or system. Also, providing multicast as an application service instead of a network service, makes the deployment in inter-domain systems much easier,

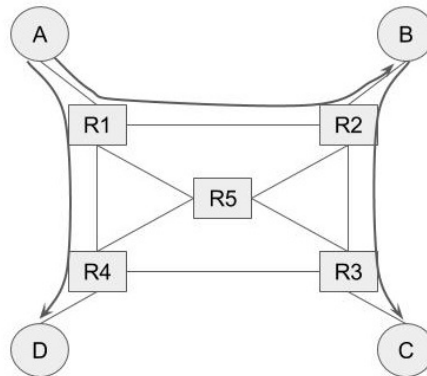


Figure 2.9.: Application-Layer Multicast delivery, adapted from Source(s): [4, 43, 45].

as the necessity for different ISPs to employ multicast solutions at the same time vanishes, and multicast comes as the result of direct peer unicast connections.

Now, as Application-Layer Multicast is performed by peers themselves, often according to a specific objective and not having a knowledge of the physical topology itself, it becomes clear that it would be almost impossible for ALM to be as efficient a solution as IP Multicast. ALM implementations often result in multicast trees with lower performance. To the previous point, such implementations are often evaluated according to a few characteristics:

- **Quality of data path:** Measured by link stress (which is the number of identical copies of data that go through a certain link [44]) and link stretch (the length of the path from any source/forwarder to destination [4]);
- **Control Overhead:** A metric that directly affects the scalability of the system. The overhead of controlling peers (both entrances and departures, as well as any distribution tree mutations) should be as low as possible taking into consideration that the number of subscribers to any session tends to be high, and an high control overhead would result in a choking point to the system, impeding its scalability;
- **Overlay Robustness:** As the overlay is dependent on the peers themselves, it is important that it be prepared to deal with a typically high churn (the number of entrances and departures to the network). A robust overlay should be able to quickly deal with peer departures, wherever they may occur. A rich connected graph with many existing connections between several different peers using different paths usually leads to a system that rarely fails.

Taking into consideration the different characteristics of Application-Layer Multicast, its advantages and drawbacks, also its various implementation challenges, it is important to analyze some illustrative research works in this area.

2.3.2 Illustrative ALM Works and Approaches

TOMA

Two-Tier Overlay Multicast Architecture (TOMA), [12, 37], is an architecture developed with the intention of, not only, providing a fast and scalable multicast system, but also one that makes sense for the Internet Service Providers as it considers a profitable service model for ISPs.

The way the multicast overlay structure has been envisioned is by constructing an overlay network, named Multicast Service Overlay Network (MSON). This overlay network is built by service nodes strategically placed along the network (by the ISP), and forwards traffic among these nodes according to the best multicast tree that can be constructed by analyzing traffic flow.

Knowing the way traffic flows through the network (i.e. via the service nodes), it is important to understand that peers connect themselves to the proxies into the MSON and form clusters around these proxies, forwarding data among themselves as well. In other words, peers form various multicast trees per session, one per each proxy regarding the peers in said proxy. Figure 2.10 provides a visual representation on the way the architecture has been designed with peers connecting themselves (and forming distribution trees) to the available proxies and these mentioned proxies forming an overlay network with unicast connections.

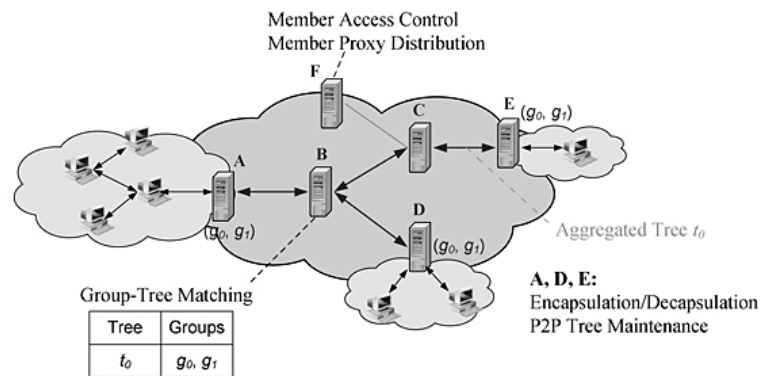


Figure 2.10.: TOMA architecture, Source: [12]

With the base model for a multicast session presented, the authors also address the scalability of the system, a crucial factor for any architecture with such a purpose. As seen in Figure 2.10, only one peer per cluster is connected to each proxy, and, as peers form distribution trees for each cluster, it is ensured all receivers get the data in the session and also that, once data is in the peer directly connected to the proxy, delivery, in a way, is no longer a concern for the system. However, the overlay network MSON has been built

with the intention of being able to deal with not only a large number of users but also a considerable amount of separate multicast sessions, which, theoretically, could result in as many ways to distribute data through the MSON, as different sources and destinations can result in different distribution trees within the MSON.

Overlay Aggregated Multicast Protocol (OLAMP) has been developed with the intention of managing the before mentioned issue. The basic idea, is to have as few different multicast trees in the MSON as possible, by having multiple groups/sessions sharing the same delivery tree within the MSON. Should there be no appropriate trees available in the beginning of a new multicast group a new tree is to be established, otherwise an existing tree will be replicated in order to minimize what can become a large number of multicast trees, resulting in extremely long (and unfeasible) forwarding tables, slowing lookup speed.

Overcast

Overcast, [33], is a not so recent approach on the creation of an overlay multicast network. The authors aim to build a scalable and reliable multicast system, capable of constructing efficient and adaptable data distribution trees.

The proposed system allows data to be sent only once to many destinations, as packets are duplicated only in the specific nodes where resources are optimized. This concept is paired with the system's capacity to cache data and create replicas along the network and so, archive content for a predetermined amount of time, allowing peers/users to access content with some delay. The system's architecture contemplates:

- Central Source: Is the single source for the distribution tree as this system implements single-source multicast;
- Overcast Nodes: These nodes form the overlay network. They are strategically placed along the physical network and are able to deal with changing conditions by employing different connections to other nodes. In other words, they are able to adapt the distribution tree at any point, a distribution tree rooted at the central source. Overcast nodes are also capable of storing data, as mentioned previously;
- Clients: Placed along the physical network, connect to the nearest overcast node.

As the Overcast implementation relies solely on the presented entities, with the distribution trees being generated only considering the necessary nodes, and while an argument could be made for the system's inherent inefficiency due to its lack of knowledge of the underlying network topology and state, overcast presents a number of advantages:

- Deployment can be performed incrementally, and so, additional nodes can be put in place as the increase in usage happens;

- Gains in performance happen when different users are accessing content at the same time, but also, when they access content at separate times, as the archiving tool allows content to be placed and accessed at a later stage.
- The distribution tree can be easily adaptable to the ever changing network conditions, should the system detect a given connection between two (or more) specific overcast nodes to be under performing, the distribution tree can (most times) be adapted to not include that specific connection, relaying traffic through other paths.

OMNI

Overlay Multicast Network Infrastructure (OMNI), [34], considers the creation of a two-tier infrastructure, directly aimed at optimizing data delivery for media-streaming applications. This infrastructure is not unlike the one presented in Section 2.3.2, but takes another very specific element into consideration in the way the distribution tree is created and managed, which is the weight of a Multicast Service Node (MSN).

The overlay infrastructure consists of two entities: a set of service nodes, MSNs, distributed along the network (in non-specific locations) and the end-hosts (which may be either the sender or one of the receivers). MSNs organize themselves in order to form a multicast distribution tree that is able to reach all nodes with end-hosts in their subscription list, and the end-hosts sole job is to subscribe to the closest MSN, requesting access to the multicast session in place. Figure 2.11 represents the developed architecture/infrastructure.

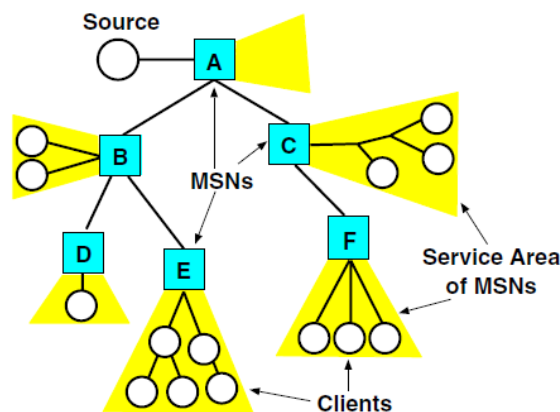


Figure 2.11.: OMNI architecture, Source: [34]

Taking into consideration Figure 2.11, it becomes clear each MSN has a set of clients (end-hosts) appended to it (i.e., in its subscription list). At this point, it is important to note that the distribution tree formed by the MSNs is independent of clients, i.e., there is

a distribution tree for the MSNs alone, and then, each MSN delivers data to its subscribers as it sees fit, either via a secondary distribution tree or via a set of unicast connections to each end-host.

Despite the fact that MSN-Client delivery is performed under a different distribution tree from the one MSNs build for themselves, this internal distribution tree formed by the MSNs considers the subscriber list of each MSN. Knowing the OMNI architecture was developed in order to optimize media-streaming applications, the authors elected to measure quality according to two factors:

1. Access load experienced by the streamers: Addressed by the overlay multicast architecture which relieves the streamer (source) from sending an unsupportable amount of data to the network;
2. Jitter from end-to-end path: Addressed by organizing the overlay paths in such a way that low-latency paths are elected when possible.

The presented quality-measuring aspects as well as the way they are addressed are insufficient for the system to operate as well as possible. To this end, in the formation of the internal distribution tree (the distribution trees formed by the MSNs), MSNs consider the weight of each MSN, i.e., the number of subscribers. With the objective of minimizing the latency and jitter to the entire client set, MSNs with a larger client set are considered to be more important than the ones serving a smaller number of clients. This determines that higher weight MSNs take precedence in path selection, i.e., they are reserved the best paths.

Scattercast

Scattercast, [36], represents another form of creating a two-tier infrastructure, but one which tries to take advantage of the benefits of the abstraction the application-layer provides, while also trying to revert to IP Multicast concepts when they are available and possible. This approach consists on the deployment of a set of strategically placed network agents, called Scattercast Proxies or SCXs, which will form an overlay network, of unicast connections, for content distribution. Then, clients will locate nearby agents and access the desired session via that agent. Figure 2.12 provides a visual representation on the authors' envisioned architecture.

Much like the architectures previously presented, the concept for Scattercast relies on data being forwarded through a distribution tree formed with the network agents, SCXs, and then, each SCX is to deliver one replica to each client. However, Scattercast is different in many ways:

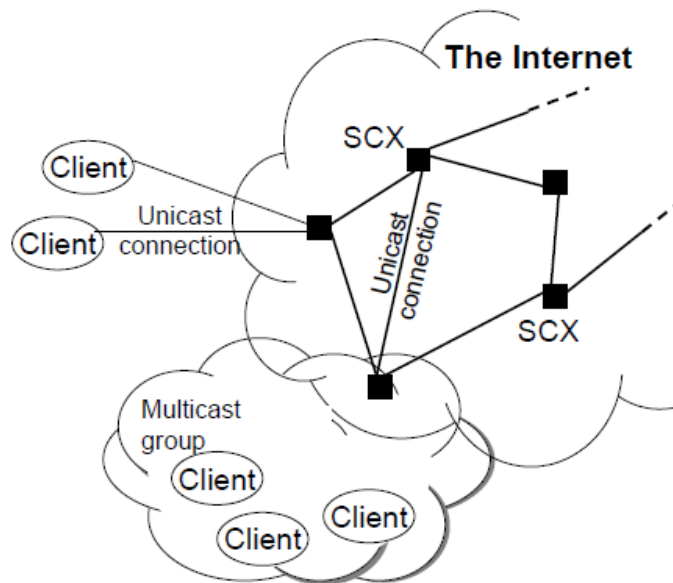


Figure 2.12.: Scattercast architecture, Source: [36]

1. While clients can tap into the session via unicast connections to the SCX and can also form distribution trees among themselves (much like in Section 2.3.2), Scattercast can even take advantage of the creation of multicast groups at the network level (IP Multicast), where they are available;
2. Scattercast allows multi-source multicast sessions, as SCXs are capable of including application-specific rules for data streaming, and so, determine who is allowed to send or receive data into/from the session.

Knowing Scattercast shares the same two-tier logic of other presented architectures, but understanding it is structurally different in the way it operates, another factor to take into consideration is the way multicast sessions themselves operate. With Scattercast, a single multicast session can enjoy multiple independent data streams, each with their own transport requirements, be it the least possible latency or reliable delivery, for instance. This notion of separate data channels for the same session introduces a huge advantage for Scattercast when compared to other systems, as the operational uses for this extension are immense.

Finally, it is important to consider the way the different SCXs interact between themselves. In order to provide support for the system and to efficiently distribute data through the overlay, an inter-SCX communication protocol was introduced, *Gossamer*. This protocol is responsible for building efficient distribution trees, as well as their maintenance. It builds a mesh structure from the unicast connections between the SCXs, and, on top of this mesh,

runs a routing protocol to determine source-based distribution trees, which accounts for two specified advantages: *i)* The generated mesh provides a more resilient system, as failures are dealt with by simply routing around the failing node; *ii)* The routing mechanisms provide ways to detect looping in the distribution paths, and so, reducing link stress.

ISP Collaboration for Traffic Reduction

The obvious use of the presented technology and systems has lead many different applications to adopt systems of this type. In doing so, application-generated traffic into the network has grown immensely, resulting in an obvious increase in the service provider's costs. This is fueled by the fact that most approaches create overlay networks with complete disregard and lack of knowledge of the underlying topology, resulting in inefficient use of resources [10, 11] such as data passing multiple time through the same network link, data crossing autonomous system boundaries when the same result could be achieved within the same system domain, etc.

With this inefficiency of resources use in mind, a number of studies are being made aiming the inclusion of the ISP in these systems, adding an entity with full knowledge of both the underlying network topology and state. With this inclusion, the application-created routing system can be made to consider not only the peers (both service nodes and end-users/clients) but also the network topology, becoming able to improve the system's performance.

In [11], the authors presented an approach that considers a topology-aware overlay multicast system. However, unlike most solutions, the goal is not solely to optimize network efficiency by reducing link usage and inter-domain data flow, but to do so while maintaining high network performance as a live-streaming scenario is to be put in place, making data delay a crucial factor.

The presented solution relies on knowing the network cost between any two nodes, as the topology is considered in the overlay formation. The authors construct the overlay with two kinds of edges, primary and secondary: *i)* Primary: Edges created with preferentially nearby peers, where the cost to the network is low; *ii)* Secondary: Edges connected randomly within the network, regardless of cost to the network.

The two types of edges considered allow for the network to operate at maximum efficiency when primary edges are in place, but also allow for unchoke mechanisms to be quickly put in place, should the primary edges (peers) fail. This mechanism, when a peer is determined to have failed, allows the system to notify the affected peers to begin using secondary connections until an adapted overlay can be formed, and restart operating under maximum efficiency.

This approach is one of many that consider the network topology in their routing algorithms, which constitutes a huge gain for the ISP, as otherwise, overlays formed without

any consideration for the underlying topology, are usually terribly inefficient in their routing. The addition of the ISP to the system, results in a scheme beneficial to all, as it leads to a reduction of the operating costs on the side of the ISP, and collaboration protocols between these entities can lead to improved overlay performance, as will be shown later in this document.

SYSTEM ARCHITECTURE AND DEVELOPED MECHANISMS

The objectives of the present work, as mentioned before, contemplate the implementation of an overlay multicast network which is adaptable to different usability contexts. With this in mind, the current chapter presents the planned architecture for the devised system, along with several associated mechanisms.

First, an introduction will be given regarding the different entities that make up the proposed system, namely peers, central node (which acts as the coordinator for any and all multicast sessions) and the collaborative service, Section 3.1. After a description of each entity, different ways of independently (i.e., without intervention from the collaborative service) constructing the multicast trees will be presented, namely the used metrics for upstream peer selection, Section 3.2. Next, some collaborative mechanisms will be described and explained, where the ISP, via the collaborative service, will cooperate with the central node to attain specific purposes, Section 3.3. Finally, the concept of extending the overlay to multiple autonomous systems will be addressed in the final section, Section 3.4.

3.1 GENERAL ARCHITECTURE

On the conception of the general architecture for the proposed system, it is important to have a good understanding of the goals that this work aims to achieve and how the different entities must be contextualized around those specific goals.

With the objective of developing an overlay multicast system capable of assuming different usability contexts, it is essential that distribution trees be easily reconfigurable both in their construction and in their management. Now, the decision on the usability context to put in place (i.e., which type of distribution trees will be formed by the peers) can be made by the session's sender (as will be detailed later) but can also be influenced by the ISP, taking into consideration that only it knows the state of the network itself at any given time. In order for this collaboration to be possible, a communication channel has to be considered between the ISP and the overlay's manager.

Taking into consideration the generalized but more practical view on goals to achieve, Figure 3.1 provides a basic visual representation on the devised conceptual architecture,





	Router
	Sender
	Receiver/Forwarder
	ISP Activated Forwarder
----	Communication Channel, ISP - Central Node
.....	Communication Channel, Central Node - Peer

Table 3.1.: Entities and interactions of Figure 3.1

which includes the entities mentioned in Table 3.1, further described in the following sections.

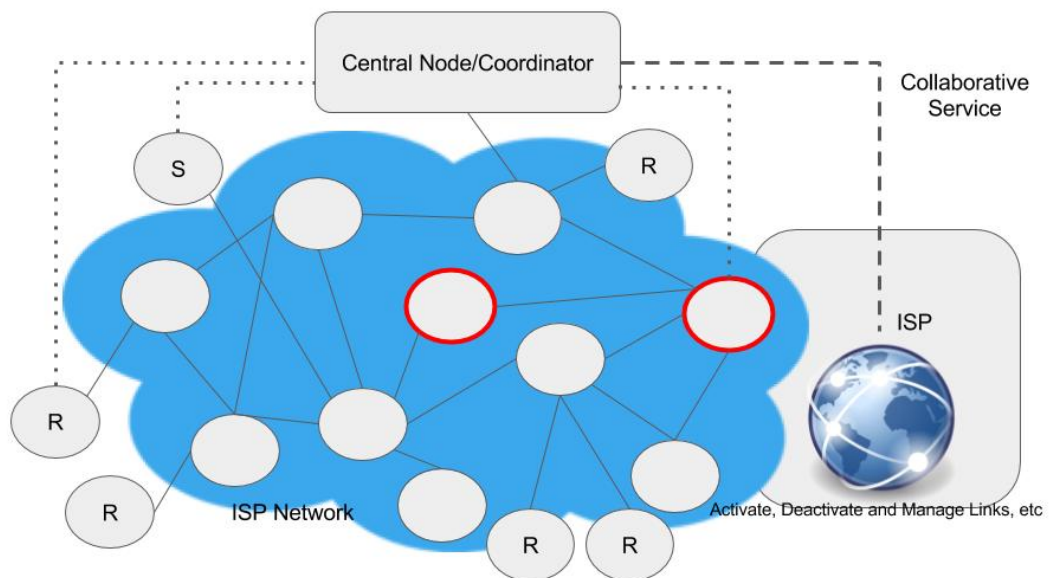


Figure 3.1.: Conceptual Architecture

3.1.1 Central Node

In more detail, the central node will be the coordinator for any and all multicast sessions. At the current point in this section, Figure 3.2 provides a visual representation on the way the central node has been structured (thus far, without the inclusion of the collaborative service).

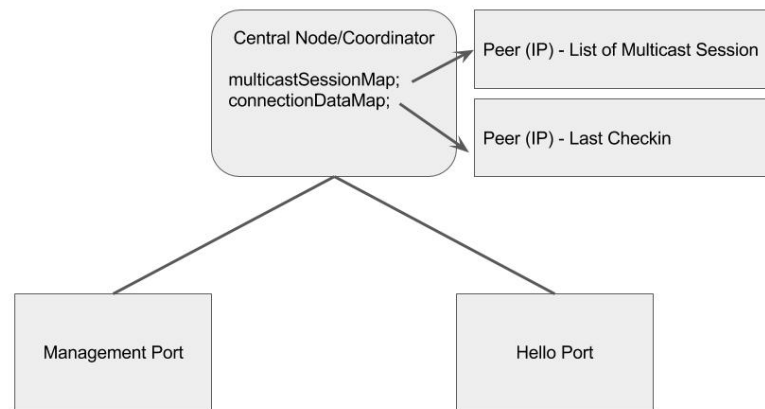


Figure 3.2.: Conceptual Central Node view

Observing Figure 3.2 it is easy to understand that the central node integrates the following elements:

- **Multicast Session Map:** This map connects each peer to the list of multicast sessions it is involved in. In the particular case of senders, they can only participate in the session they are streaming to; Receivers can participate in as many multicast sessions as they wish (always as receivers of content);
- **Connection Data Map:** The central node uses this map to keep track of the live-state of peers. Time stamps of Hello Messages are stored in this data structure, and a worker notifies the central node to remove any specific peer from all sessions in case the mentioned peer's last time stamp exceeds a predetermined amount of time;
- **Hello Port:** This is the port peers use to send their respective Hello Messages in order to be allowed to remain in the system, as it employs a soft-state protocol;
- **Management Port:** This is the port peers use to communicate with the central node, exchanging management messages, with some of the more important being:
 - *Create Multicast Session:* Any peer can create a multicast session (and be immediately set as the sender for that session) and begin streaming data upon receiving the central node's authorization;
 - *Join Multicast Session:* This is the message receivers wishing to join a multicast session send the central node, asking for information on the session with the desired sender;
 - *Join Peer Notification:* This message is absolutely critical to the proper simulation of multicast distribution trees by the central node. It is the message peers

send the central node when they make a decision on which peer they append themselves to, making it possible for the central node to perform accurate representations of multicast sessions, a concept that will be detailed further in this document, in Section 3.1.1.

In summary, the central node, as described thus far, is already instrumental in facilitating the system's operation, as it:

1. Filters the whole system for peers that have left, and notifies the affected peers, for instance, the peers that were receiving data from the departed peer and would, otherwise, be left without the data being streamed into the session. The upstream peer from the peer that left is also notified to update its downstream list, meaning, removing the peer that left.
2. Maps every peer to the session it is involved in, and so, with the information exchanged with peers, is able to maintain an accurate and updated state on the distribution tree of each multicast session.

Multicast Session Operation

In section 3.1.1, the central node has been described as the brain of the operation. The concept of multicast session is an intricate part of the central node, as it is the structure that combines the information that the central node obtains via its communication with all the separate peers involved in each session.

To the multicast session, peers can either be the sender (for simplification, it is assumed that there is only one sender per session) or receivers. Receivers can also act as forwarders in case they represent a better performance option than a direct connection to the sender. This happens when peers are applying their QoS tests to other peers, and the obtained results to other receivers represent improved efficiency (which will be further detailed).

It has been established that any peer that wants to create a new multicast session informs the central node, which then creates the session and places the mentioned peer as the sender. On the central node's part, this allows for the creation of a new Multicast Session instance, and so, including it in its records. With this record having been created, when a receiver wishes to join a given session the process is a bit more complex and will assume different metrics according to different usability contexts, still, it follows similar guidelines:

1. The joining peer messages the central node notifying it of its intent to join the desired sender's session, identified by its IP Address;
2. The central node, as all peers perform this process, knows every peer in the session, and replies with the IP address of every connected peer;

3. The joining peer, then performs the appropriate QoS tests with every peer in the session, and elects the one it wishes to join, meaning, elects its upstream peer;
4. The joining peer, then, notifies the Central Node of the performed choice.

The previous (simplified) description of actions performed by a peer that wishes to join a specific session, in particular, step 4, allows the central node to keep a replica of the current distribution tree. This has to do with the fact that, from these notifications, the central node now knows both the joining peer and the elected upstream peer. This allows for a graph representation of the multicast distribution tree, as seen in Figure 3.3, which presents an illustrative example:

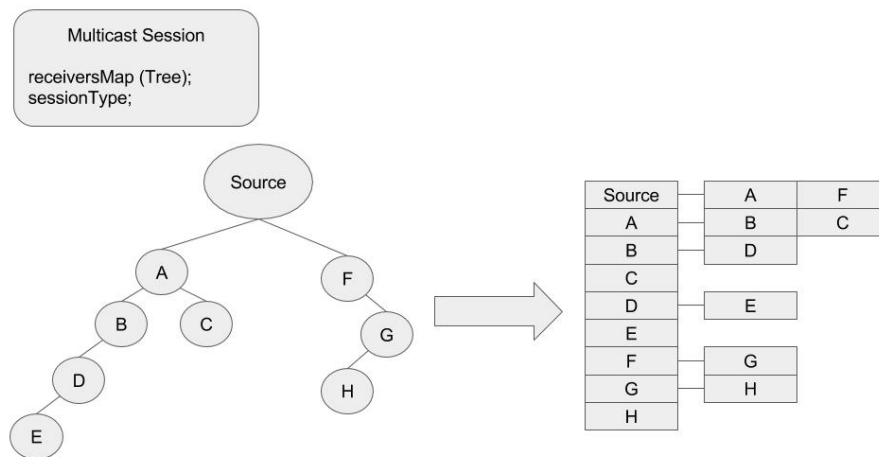


Figure 3.3.: Distribution Tree example, Central Node's Graph Representation

- The Source is only streaming data to Peers A and F;
- Peer A's upstream is the source and its downstream list are peers B and C;
- Peer B's upstream is peer A and its downstream list is peer D;
- Peer C's upstream is peer A and has an empty downstream list;
- Peer D's upstream is peer B and its downstream list is peer E;
- Peer E's upstream is peer D and has an empty downstream list;
- Peer F's upstream is the source and its downstream list is peer G;
- Peer G's upstream is peer F and its downstream list is Peer H;

- Peer H's upstream is peer G and has an empty downstream list.

As previously discussed, an absolute requirement for the proposed overlay system to perform as desired, was that the distribution trees were to be easily reconfigurable and adaptable as different usability contexts may be put in place if and when it is deemed necessary. This graph representation allows an easy manipulation and checkup of the multicast distribution tree, and comes as a result of the existent communication channel between peers and the central node, adding no communication overhead.

3.1.2 Peer

At this stage, with the central node and its role in the overlay network having been presented, it has to be made clear that the overlay itself is composed by the peers present in each session. With this concept in mind, a peer can be either the sender of the session or one of the receivers. Figure 3.4 represents the structural logic of the peer entity.

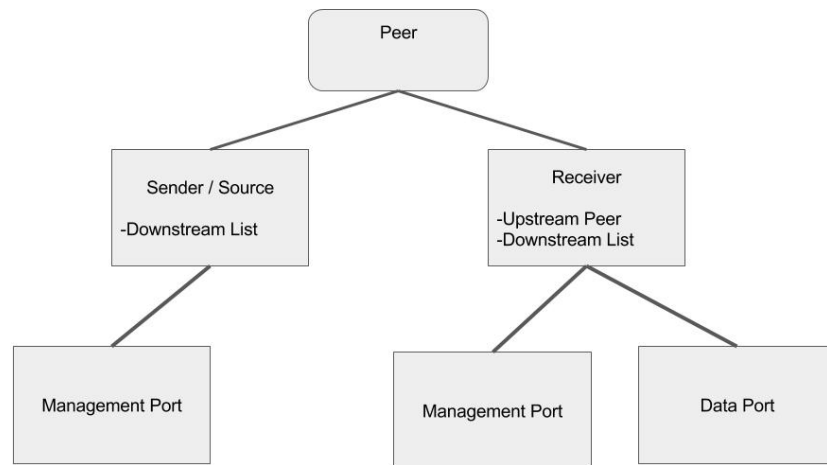


Figure 3.4.: Conceptual Peer View

Sender

Previously in section 3.1.1, in a simplified way, it has been stated that any peer that wishes to create a multicast session (and so, be placed as the sender for that specific session), communicates with the central node to perform that request and obtain its reply and authorization. On the sender's part, to make this communication possible, a Management port was added, as seen in Figure 3.4.

Another fact that is represented in Figure 3.4 is that the sender contemplates as downstream list, which represents (possibly part of) the peers in the session, or at least, the peers it is directly streaming to. Once more, the sender will receive QoS probes and Join requests in its management port.

Receiver

Analyzing, once more, Figure 3.4, it becomes clear that the receiver is structurally different from the sender, despite also being a peer.

First and foremost, the receiver has not only a management port but also a data port. The data port is where the session's actual streamed data is sent to, whereas the management port is where interaction with the central node and with other peers is performed. For example, QoS probes and peer join requests are sent through this management port.

The second distinction between the sender vs receiver logic, is that receivers contain an upstream peer, which represents the peer that the receiver is getting data from, in other words, it is the peer to which the receiver is appended to. Now, should it be the case where all receivers connected themselves to the sender, there would be no necessity to keep a record as to the peer's upstream, as it would always be the sender. The reason why this information is maintained, and the reason why peers have to perform a QoS gathering process (which may follow different methodologies and approaches) with other peers in order to elect their upstream peer, is that peers may elect as their upstream, another receiver in the session, in which case, the elected peer will also act as a forwarder in the session.

So, with the notion that a receiver was also conceived to operate as a forwarder, another analysis can be performed on Figure 3.3:

- The Source is only streaming data to peers A and F;
- Peer A is receiving data directly from the source, making two copies and forwarding to Peers B and C;
- Peer B is receiving data from peer A and forwarding it to peer D;
- Peer C is receiving data from A, while sending data to no one;
- Peer D is receiving data from B and forwarding it to peer E;
- Peer E is receiving data from D, while sending data to no one;
- Peer F is receiving data from the source and forwarding it to peer G;
- Peer G is receiving data from peer F and forwarding it to peer H;
- Peer H is is being forwarded data by peer G, while sending data to no one.

The concept of having receivers perform forwarding duties allows an enhancement of the overlay's performance as paths from source-receiver may have higher delays or packet loss percentages and, in some cases, a longer path using receiver-receiver direct connections may come as a better solution, performance-wise, so traffic engineering may be put in place in similar situations.

3.1.3 *ISP Collaborative Service*

The described architecture contemplates the central node (coordinator for the overlay network) as well as the peers that form the overlay network itself. This allows for the creation of multicast sessions, as well as some decision-making on the peer side, namely, each peer is able to choose its upstream peer, as explained previously.

The addition of the collaborative service, which represents a communication channel between the ISP and the overlay network (via the central node), allows a paradigm change to the present architecture. This is due to the fact that the overlay network will now be able not only to work on top of the network but also to work with it, as the physical network itself belongs to the ISP. The previous concept is only made possible as the ISP is now able to communicate with the overlay's managing entity.

While the ISP will not be an active participant in multicast sessions taking place, it can be of great influence in the distribution tree's formation. Without the ISP, distribution trees come out of peers electing their upstream peer, however, the ISP's knowledge of the network topology allows for a number of additional mechanisms and functionalities that can be introduced in the overlay level.

The way the presented concept works is by having the ISP, via the collaborative service, provide the central node with a topology graph, which represents the whole network topology. Additionally, the ISP also provides knowledge of the access router each peer is using into the network, so that the central node may (knowing the routing protocol in place) simulate and determine the best path between any two peers. This information allows the central node to more efficiently manipulate the distribution tree's formation and management.

While it is true that this concept comes with some additional work for the central node, it brings many advantages as well. This has to do with the fact that the ISP knows not only the network's topology, but also its state. So, in case, for instance, a link is receiving too much traffic and/or losing datagram packets, or even in case the ISP knows that at a certain point in time (with traffic agreement) a given path will have increased usage, the ISP can have central node re-direct, if possible, overlay traffic and deviate from using any path the ISP wishes to protect, whenever possible, leading to a much more stable network. This traffic engineering possibility could only come with a full knowledge of the network topology and some routing level details, which only the ISP can provide.

With the central node being able to simulate the paths taking place, considering the distribution tree, sometimes, tree management and manipulation is not possible taking into account the current peers in session, i.e, the way they are spread through the network does not allow for traffic improvement. However, the ISP not only owns and knows the network topology but also is able to activate entities within it. To this point, the ISP can, at any given time, activate (and deactivate) overlay forwarders in strategic places within the network, and have traffic go through them, when they result in performance or security improvements. This allows for the deviation of traffic, sometimes resulting simply in avoiding certain parts of the network the ISP wishes to see freed, or resulting even in a reduction of the number of links the distribution tree is using.

3.1.4 Extended Central Node

The addition of the collaborative service to the architecture of the present work, implies some changes to the central node's structure, which are represented in Figure 3.5.

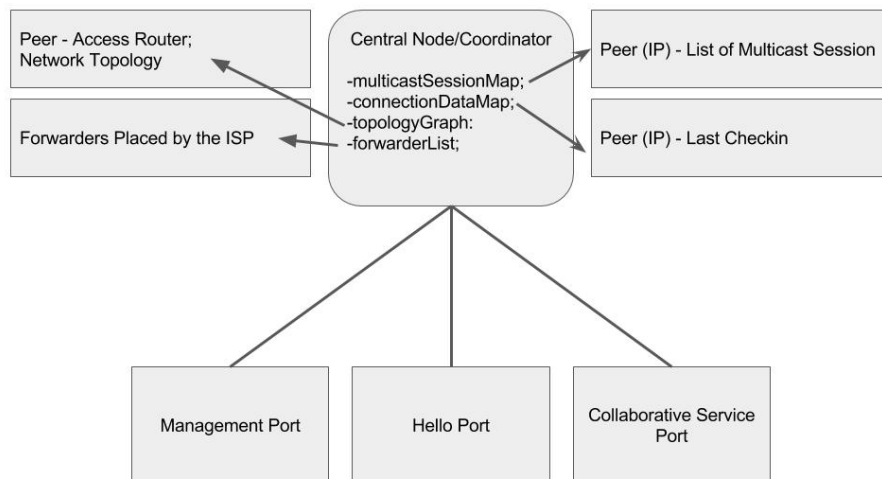


Figure 3.5.: Conceptual Central Node, Extended

The inclusion of the collaborative service provides the central node with an extended number of alternatives in the overlay management. This new version of the central node now contemplates some new elements:

- Topology Graph:
 - Actual Topology: Contains the topology layout;
 - Access Routers: Contains all the information regarding peers themselves, meaning, the access router each peer is using into the network;

- Routing level information;
- Forwarder List: Contains the list of forwarders the ISP has provided to the overlay, which, at any given time can be updated as the ISP can activate (or deactivate) forwarders as it sees fit;
- Collaborative Service Port: Whilst the central node's management port could be used in the communication with the ISP (via the collaborative service) a decision was made to create a dedicated port. Some of the exchanged messages/requests (which will be detailed later in this document) are:
 - Link Protection, where the ISP requests that the central node protects a given link, if possible;
 - Link Minimization, where the ISP requests that the central node try and reduce, as much as possible, the number of links in use, by the overlay;
 - Forwarder Activation, where the ISP notifies the central node of changes to the available forwarders and has it test multicast sessions for improvements to performance;
 - The mentioned network info (topology graphs, access routers and routing information) will also travel through this port;

Once more, in summary, the central node is now able to operate with more extensive knowledge, allowing for traffic engineering/management through the overlay. The ways the central node will perform this management will be detailed along the present document and will take many forms, namely with regards to different approaches or usability contexts, and mostly with regards to the collaborative methods that take place between the overlay and the ISP via the collaborative service.

3.1.5 *Extended Peer*

The inclusion of the ISP collaborative service to this work, in theory, should bring no change to the peer, as any interaction is made with the central node, and does not include the peer. Despite this fact, a decision was made to change the peer architecture as well, as represented in Figure 3.6, including the ISP activated forwarder as a peer to the overlay. This allows for no change to be necessary to the multicast session, as an entity, as the forwarder is seen as a regular peer.

Despite the fact that, to the multicast session, the ISP activated forwarder is a regular peer, in particular, a regular receiver, its structure is different. While it shares the fact that it has both a management port and a data port, it does not receive (or respond to) QoS probes

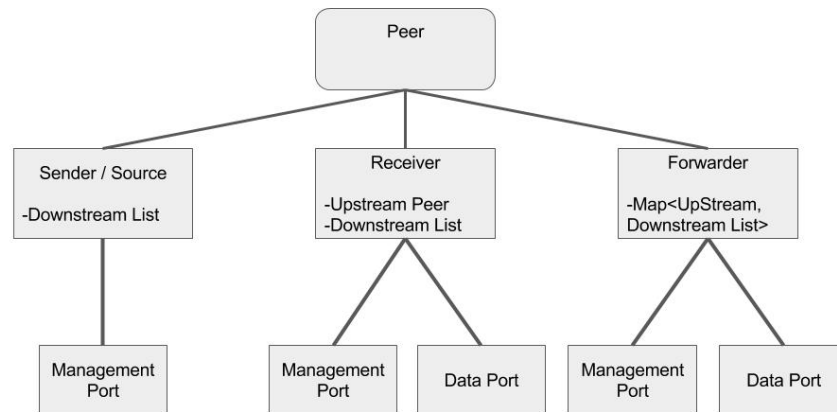


Figure 3.6.: Conceptual Peer, Extended

as it operates only under instructions from the central node, never from other peers in the session. So, the management port is used only for communication with the central node and the data port also operates in a different manner.

Normal receivers have an upstream peer, and when messages arrive through the data port, they are replicated and forwarded to the downstream list should said receiver be acting as a forwarder. ISP activated forwarders operate differently as they may be forwarding messages from various sources. What this means is that, when messages arrive through the data port, they are checked to see where they come from so that the proper downstream list for that source is used as actual forwarders are to serve as many peers as possible, which is why, as seen in Figure 3.6, forwarders have not only one upstream, but a collection of them, associated with the corresponding downstream (forwarding) list.

3.2 DISTRIBUTION TREES CONSTRUCTION

This section will explain the way multicast trees are formed according to different usability contexts. Within this purpose, two distinct examples will be provided. The first scenario will be one of cumulative minimum-delay, where peers will try to minimize data delivery delay from the source/sender to each receiving peer in the multicast session, Section 3.2.1. The second scenario to present contemplates cumulative packet loss percentages that the system will try and minimize, Section 3.2.2.

3.2.1 Minimum-Delay Approach

When peers receive instructions to form the multicast tree favouring an objective of minimizing the delay from sender to each receiver, a cumulative scenario is started where peers connecting to multicast sessions distribute themselves in such a way to make end-to-end delay as small as possible. To this point, every different entity is responsible for a set of tasks in order to make the process viable.

Following, a description is given on the logical steps the various entities involved perform.

Creating a (Minimum-Delay) Multicast Session

The first step to be taken in the creation of a multicast session, is to have the peer that will act as the source (sender) for that session request the central node to create and register a new session. Figure 3.7 illustrates the rather simple process the sending peer and the central node go through in order to establish a new multicast session.

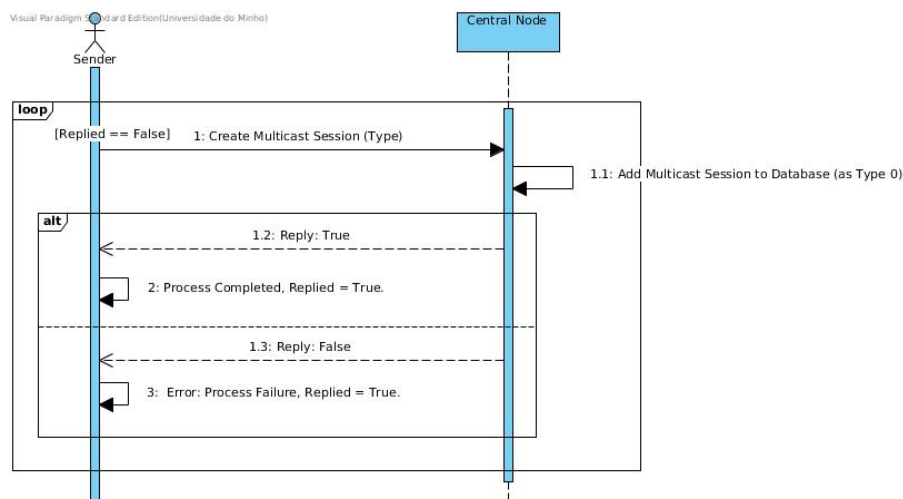


Figure 3.7.: Activity Diagram - Creating a Multicast Session

After receiving authorization from the central node, the sender then starts streaming data to the multicast session. A note should be made regarding Figure 3.7, particularly the fact that the presented loop has to do with the fact that the protocol in use is UDP, making it necessary to verify the arrival of any and all management packets.

So, in summary, the process to create a new multicast session is as follows:

1. The sender (peer) contacts the central node with a message informing of its intent to create a new multicast session, type 0 (minimum-delay);

2. The Central Node, verifies the existence of a multicast session with said peer as the sender:
 - a) If the session does not exist, the central node adds it to its database of multi-cast sessions and replies "True" to the sender, so it may start transmitting data, whenever the session's first receiver arrives;
 - b) If the session already exists, should the type of the existing session be the same as the type of the session being requested, the central node, assuming the reply packet was lost in transit, replies "True" so the sender may start transmitting data on the arrival of the first receiver. In case a session with that peer as the sender already exists, but under a different usability context (other type), the central node will reply "False".
3. The sender, having received the Central Node's authorization, starts streaming data. As this data streaming happens directly to any joining peers, it begins only after the first receiver has connected. Should no reply arrive in a predetermined amount of time, the sender will repeat the process, from Step 1.

Joining a (Minimum-Delay) Multicast Session

When joining a multicast session, a peer must know the sender's IP address, which determines the multicast session it is joining. With that, the procedure to join a session takes the following steps:

1. The joining peer sends a message to the Central Node (coordinator) asking for information on the session it wishes to join;
2. The Central Node replies with the IP Address of all the Peers involved in the session;
3. With the list of all peers in the distribution tree, the joining peer, then, messages every single peer in the reply. This serves two purposes. The first it that it allows the joining peer to request the queried peer's delay until the sender;

In this scenario, it is important to remember that a cumulative delay is considered, and so, while the joining peer requests every other peer's delay until the sender, this information has to be complemented with the delay between the joining peer and the peer it is appending itself to, which is the second purpose of the mentioned message exchange. With this in mind, an algorithm was created to:

- a) Message all peers in the session requesting their delay to the source;
- b) Measure the reply time from each peer, and add this value to the peer's delay to the source, which is done via the mentioned probe message;

- c) Select the peer which results in the minimum cumulative delay from the joining peer to the session's source;

Figure 3.8 aids in the understanding of the created algorithm.

In it, receivers R1, R2, R3 and R4 are directly connected to the sender, and receiver R5 is connected to R4. So, when R6 joins the overlay, the selection of the peer to join is: $\text{minimum}\{\text{Delay}(\text{R6-S}); \text{Delay}(\text{R6-R1})+D_1; \text{Delay}(\text{R6-R2})+D_2; \text{Delay}(\text{R6-R3})+D_3; \text{Delay}(\text{R6-R4})+D_4; \text{Delay}(\text{R6-R5})+D_5+D_4\}$.

The previous reasoning, was made according to Equation 1, where D stands for Delay, R means receiver, and j is the joining peer:

$$\text{selectedpeer} = \min_i(D(R_i) + D_{j \rightarrow i}) \quad (1)$$

4. The joining peer, having selected the peer it wishes to append itself to, messages the selected peer of this intent. In other words, requests to be added to its forwarding list;
5. The peer that received the joining peer's request, then simply adds the joining peer to its forwarding list and replies informing the joining peer that it has accepted the request;
6. The joining peer, having received the reply from the peer it is appending itself to, then opens its data port to start receiving data and sends a notification to the central node informing it of this connection;
7. The central node updates the multicast session registries with this new information, and now considers the new peer as a part of the session when other replies or requests arrive.

An important factor to take into consideration is that, at any time, packet loss may occur. So, in this process, for example, in Step 3, should no reply arrive from the central node, the peer simply repeats the process, from Step 1. Furthermore, within the whole process, all QoS probing and management messages are checked for delivery.

3.2.2 Minimum-Loss Approach

In the previous section, the usability context's goal was to provide the best possible performance for the multicast session, in terms of end-to-end delay to each peer involved. In this section a different approach is taken, as the objective is to minimize, as much as possible, the loss of data being streamed into the session. With the presented usability context in

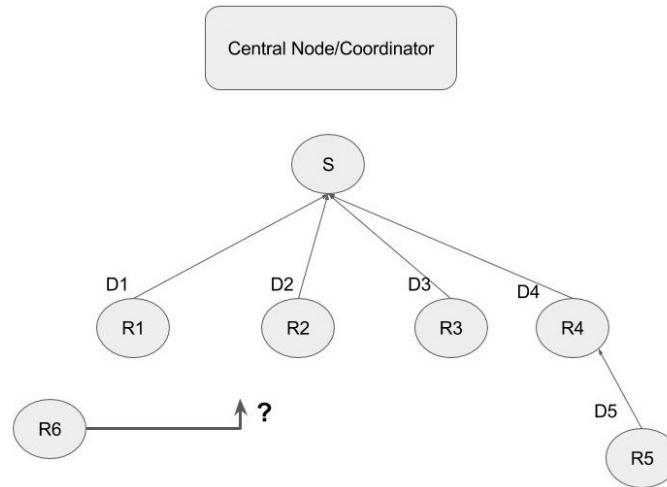


Figure 3.8.: Joining Peer Decision Example

mind, the way peers connect themselves to the session, for obvious reasons, must change. Meaning, the way each peer elects its upstream is different, as will be shown bellow.

The remainder of the present subsection will describe the way the sender creates this new type of session and also, the process each peer performs when entering the session.

Creating a (Minimum-Loss) Multicast Session

When creating a multicast session with a minimum-loss usability context, the first step to be taken is to have the peer that will act as the sender of that particular multicast session, request the central node to create and register a new session. At this point, while the logical process is the same as was presented for the minimum-delay scenario, the new multicast session that the central node registers is tagged with a different session type value, corresponding to minimum-loss in the present case. Figure 3.9 illustrates the process the sending peer and the central node iterate through in the creation of a new multicast session using the minimum-loss usability context.

The peer (sender), having received the reply from the central node acknowledging the creation of the multicast session, then begins streaming data to the session as the first peer (receiver) connects.

Now, as mentioned previously with the minimum delay usability context, packets can be lost in traffic. To this point, the process must be repeated consecutively until the case where the sender receives the actual reply from the central node, as seen in Figure 3.9, regardless of where the datagram packet may have been lost. While this is true for any approach as UDP packets are exchanged throughout the system, this scenario requires special attention as packet loss is to be measured, and QoS probing messages are not repeated when no

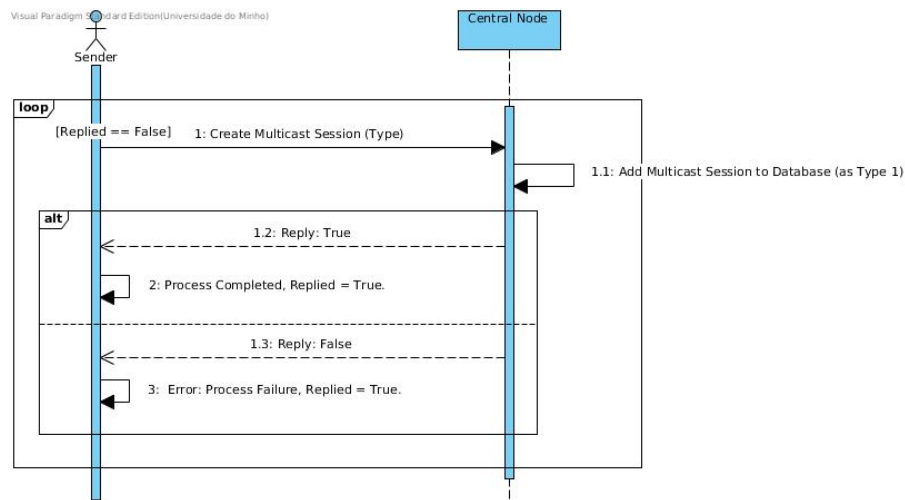


Figure 3.9.: Creating a minimum-loss multicast session

reply arrives. With this in mind, the process to create a minimum-loss multicast session is as follows:

1. The peer (sender) contacts the central node with a message informing of its intent to create a new multicast session, type 1 (minimum-loss);
2. The Central node, verifies the existence of a multicast session with said peer as the sender (regardless of type):
 - a) If such a session does not exist, the central node adds it to its database of multicast sessions and replies "True" to the sender, so it may start transmitting data;
 - b) If the session already exists, should the type of the existing session be the same as the type of the session being requested, the central node simply assumes the packet was lost in transit and replies "True" so the sender may begin transmitting data; (Note: Should a session with that peer as the sender already exist, but under a different usability context, the central node will reply "False".)
3. The sender waits for the central node's reply:
 - a) Should the answer from the central node arrive, it will simply start transmitting data to the session on the arrival of the first receiver and the session will operate normally;
 - b) Should it be the case that no answer arrives, the peer simply repeats the process by jumping back to 1 until a maximum of five times. The time the peer waits for the central node's reply is five seconds.

With the connection process being complete, the session is either activated and the sender starts transmitting data or it is removed as the sending peer leaves the system.

An important reference to be made is the fact that, this packet delivery validation is only performed in such management data exchanges, once the session is active, the session data (i.e., the data being streamed) is not confirmed for arrival in each peer. This work assumes that multicast distribution trees are used to transmit real-time data flow applications using the UDP protocol.

Joining a (Minimum-Loss) Multicast Session

As before, when joining a multicast session, the joining peer must know the sender's IP Address, which identifies the session it wishes to join. However, the joining process now differs from the one previously detailed in section 3.2.1 particularly in the way QoS probing is performed. With this new management scheme in mind, the joining procedure is as follows:

1. As any multicast session can now be of different types, it is crucial that the joining peer is made aware of the usability context in use so that it may elect the peer it will append itself to in an according manner. The peer, having received the reply from the central node regarding the type of session in place, should it be 0 (Minimum-Delay), simply resumes the standard minimum-delay connection setup, detailed previously, should it be 1 (Minimum-Loss), moves on to step 2, starting the minimum-loss connection setup process.
2. The peer, messages the central node requesting information on the peers already in the session. As this is a management interaction, requires packet delivery validation, meaning, the peer cyclically performs this request until a response finally arrives. The delay between one request and the next being performed (should no reply arrive) is 5 seconds.
3. On each request, the central node replies the requesting peer with the IP Address of all Peers involved in the session. At this point, the central node performs no validation at all, simply replies to every single request, any needed delivery validation regarding this type of message happens on the peer's side.
4. The joining peer, having received, from the central node, the IP Addresses of all peers in the session, then begins the best peer selection process:
 - a) At this point, for each peer in the central node's reply, the joining peer must gather QoS information regarding the average packet loss from such destination. It is important to understand that the aim is to reduce packet loss in the streamed data, meaning, tests must be performed downstream through the distribution tree, i.e., packet loss is measured from the tested peer to the joining peer. As such, an auxiliary answer manager was developed in order to store each peer's

loss percentage from the session's sender (if it is the actual sender, that value is zero). Then, QoS probing begins and the answer manager increments the number of obtained answers every time one arrives. The joining peer will request each peer in the session to perform the tests, and each peer will reply with a burst of datagram packets, which will later be used to determine the packet loss percentage.

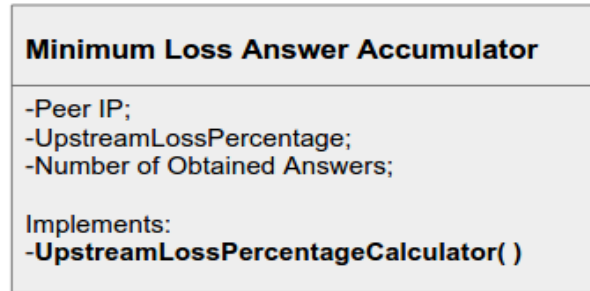


Figure 3.10.: Minimum Loss Answer Accumulator

- b) The joining peer, having performed the necessary QoS tests to each peer returned in the central node's reply, must then elect the peer which it will append itself to. This minimum-loss usability context is performed considering a cumulative method of calculating the peer's loss percentage from the sender, which means, that the final decision on the elected peer considers both loss percentage from the joining peer to the appending peer as well as the appending peer's loss percentage to the session's source. This process is translated in the minimum loss answer accumulator, which implements an upstream loss percentage calculator method that uses Equation 2 in order to determine the peer to join, where j is the joining peer, i is the peer being tested, and P represents packet loss percentage.

$$selectedPeer = \min_i(1 - ((1 - P_{j \rightarrow i})(1 - P_i))) \quad (2)$$

5. The joining peer, after electing the peer to append itself to, now has to establish a direct connection between itself and the peer to append. To do so, it sends the selected peer a join message. This joined peer, having received a join message, adds the joining peer to its downstream list, replies "True" to the joining peer and starts forwarding data to the joining peer. Once more, this is a session management interaction between these two peers, packet arrival must be validated, and so, if the joining peer does not receive the expected reply from the appended peer, it simply repeats the direct connection establishment process.

6. Should the connection between the joining peer and the appended peer have been successfully establish, the joining peer needs only to notify the central node of such connection, as it is essential that the central node maintains an updated state on the multicast tree. To this point, the joining peer messages the central node of the performed connection, a notification that the central node replies, as this is a management interaction and must be validated. The joining peer, once more, repeats the process until a confirmation arrives.

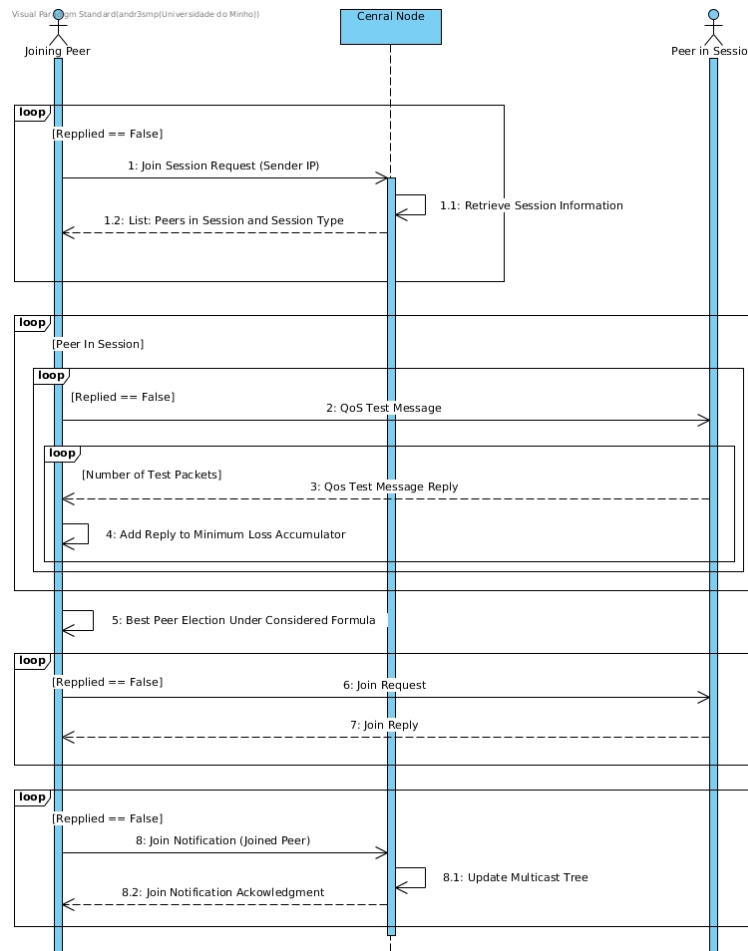


Figure 3.11.: Joining a minimum-loss multicast session (Simplified)

Figure 3.11 represents a simplified (some steps have been joined) version of the described process, but illustrates rather accurately the way the process is performed. In step 5, where the best peer is elected, Equation 2 is used to perform the determination.

3.3 COLLABORATIVE METHODS / APPROACHES

This section will present the developed collaborative methods between the ISP and the overlay network. The importance of such a collaboration has been addressed previously in this document and various approaches can be taken in the establishment of such a cooperation. This section will detail three illustrative examples. First, Link Protection will be addressed, both passive and active modes, Section 3.3.1. The second approach to be addressed will be Link Minimization, where the ISP requests the central node to try and reduce the number of ISP links the overlay is using, Section 3.3.2. These two first approaches, do not actively involve the ISP, it simply performs the requests. The following methodology, ISP Forwarder Activation, enables the ISP to activate forwarders along the ISP network, elements activated in order to optimize the overlay's impact in the network, Section 3.3.3.

3.3.1 *Link Protection*

The first collaborative method to be established between the ISP and the central node is the capability to protect the underlying network. This type of protection of the physical network itself, is achieved through the avoidance of certain links from the overlay's operation.

The idea is to avoid the introduction of traffic to any links in the topology that the ISP wishes to protect. However, in order for such a system to work, the central node needs to be provided with both the network topology and the access routers each peer is using, as well as the necessary routing information. With that in mind, the ISP, via the collaborative service, is able to provide the central node with the required information. Moreover, at any point, the ISP also knows the state of the network and may request that the central node protect certain links by distributing traffic, where possible, through other paths. This traffic redirection is made by changing the way the multicast distribution tree is constructed, in other words, by having peers append themselves to other peers, when possible.

The central node must, then, be able to determine whether or not the path between any two peers involves any of the links to protect. To aid in path determination between two peers, a graph library was used, as will be shown in Section 4.1.1.

As far as link protection goes, a decision was made to implement two different strategies, one which simply contemplates the protection of a given link by impeding new connections using that link, and another which actively attempts to stop already established connections between peers whose path involves the links to protect. They were called passive and active link protection, respectively, and are described bellow.

Passive Link Protection

Passive link protection, as the name suggests, does not directly influence the multicast distribution tree, at least, as far as connections that have previously been established are concerned. Passively protecting the network topology means that any new peers that wish to join the multicast session have their list of possible connections filtered in such a way as to avoid passage through certain areas of the network, even though this may impede the multicast tree from reaching maximum performance for the usability context in question.

The way the system operates is according to the following steps:

1. The ISP, via the collaborative service, messages the central node notifying it of the intention to protect a given link;
2. The central node, adds that link to the list of links to protect;
3. When a given peer messages the central node with a join session request, whereas in a normal scenario the central node would simply reply with the address of every single peer involved in the session, in this case the process is a bit more complex:
 - a) The central node uses its connection index to determine the access router the joining peer is using;
 - b) The central node, for each peer already in the session, uses the graph library presented in Section 4.1.1 to determine whether or not the path between said peer and the joining peer includes any of the links to protect. Should it include the link, that peer is not included in the set of peers the joining peer will receive as a reply, otherwise, it will be added to the list and will be one of the possible connections;
 - c) The central node then replies the joining peer with a list of the peers to whom the path does not include the protected link;
 - d) Should the paths to all possible peers in the session use the links to protect, the reply goes into default mode, and simply includes all peers in the session, meaning that it is impossible to protect the network in that specific case;
4. The joining peer, then, resumes the normal connection process, contacting the peers included in the central node's reply and appending itself according to the usability context in place.

Figure 3.12 provides a simplified visual representation of above the described process.

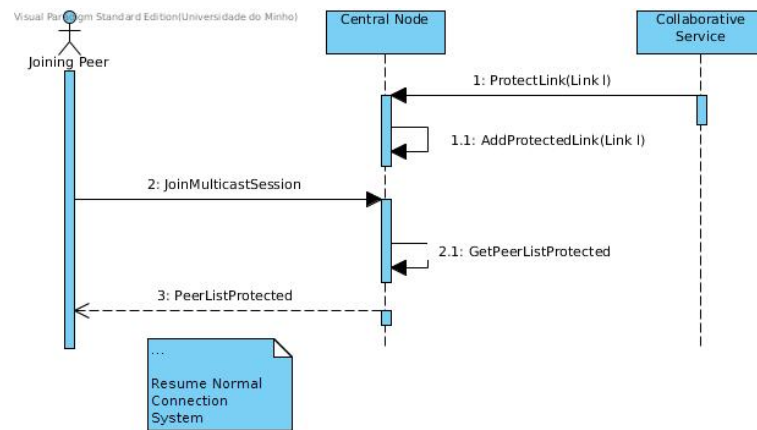


Figure 3.12.: Passive Link Protection

Active Link Protection

Active link protection is a much more intrusive and costly type of protection. Whereas in the passive protection mode, there was never a necessity to restructure the multicast tree, in an active protection scenario, existing connections are disrupted should the path between any two connected peers include the link to protect.

This type of link protection, obviously, is much more aggressive, not only to the central node, which has to perform quite a lot more operations in order to determine the necessity to have the multicast tree reassemble itself, but is also rather more costly to the overlay itself, as it destabilizes the multicast tree, and therefore introduces extra traffic into the network, with peers flooding it with requests to the central node and to each peer in order to determine the best solution to re-append themselves to.

As far as the algorithm is concerned, in order to reduce the computational weight of the process, the way the system protects the network is:

1. The ISP, once more, via the collaborative service, messages the central node notifying it of its intent to protect a specific link;
2. The central node, adds that link to the list of links to protect;
3. The central node, for every single peer in the multicast session, tests if the path from such peer to each of its children (peers that are in the downstream list of the mentioned peer) includes the link in question. If the link to protect is not included, the central node simply moves on to the next peer in the list. However, if the path does include that link, a few decisions had to be made:
 - a) The peer that is receiving data that is going through the link in question, is notified by the central node in order to reconnect itself to the multicast session. The reason is simple, instead of determining a new parent for this peer (which

would be much more complex, in an algorithmic sense), the idea was to have the affected child reconnect itself as the system will default to the passive protection mode, where the child's current parent will not be included in the list of possible parent peers due to the fact that the link has been added to the protected links list;

- b) The children of the notified peer (which, itself, was a child of another peer) are not notified by the central node, they simply wait for their parent to reconnect itself to the session. This decision was made in order to try and reduce the flooding of requests that would result by having every single peer in the tree branch try and reconnect, in other words, this decision provides for a more stabilized multicast tree as well a more stabilized network;
4. The affected peer then simply resumes the normal connection mode into the multicast session.

3.3.2 *Link Minimization*

In this current section another way to protect the network is documented, however, this protection is achieved via a different format. Whereas before, the goal was to protect a certain link (or group of links), with Link Minimization the goal is to reduce, as much as possible, the number of ISP links in use per multicast session, and so, also reducing the number of packets in the network.

Now, as protection is performed under a different logic, it is important to understand that, before, protection was guaranteed, where possible, by having the central node disconnect specific peers in the multicast session in order to, then, manipulate the way they established their new connection (upon re-connection). Under the new paradigm, the central node will simply perform all the work, and then notify each peer in the session of their new upstream and downstream peers.

In this link minimization approach to protecting the network, performance is not taken into consideration and, as such, this procedure works the same way regardless of the metric each peer uses in order to determine its upstream peer, which is why, when this method is activated, the peer's choice is removed, and it appends itself to whomever the central node determines. Each peer's downstream list is also calculated by the central node.

So, in this link minimization method, so that the central node computes the appropriate peer associations, the following steps are taken:

1. Creation of a complete graph/map, Section 3.3.2;
2. Peer indexation and cost association, Section 3.3.2;

3. Calculation of a Minimum Spanning Tree, Section 3.3.2;
4. New Multicast Tree, Section 3.3.2.

With the presented steps in mind, the final aim is to have each peer in the session receiving the intended data, but with the overlay using the lowest possible number of network links.

Activation of Parity Maps

The first step in order to achieve the desired goal is the creation of a complete graph where each connection’s cost is represented with the shortest path from each origin to each destination.

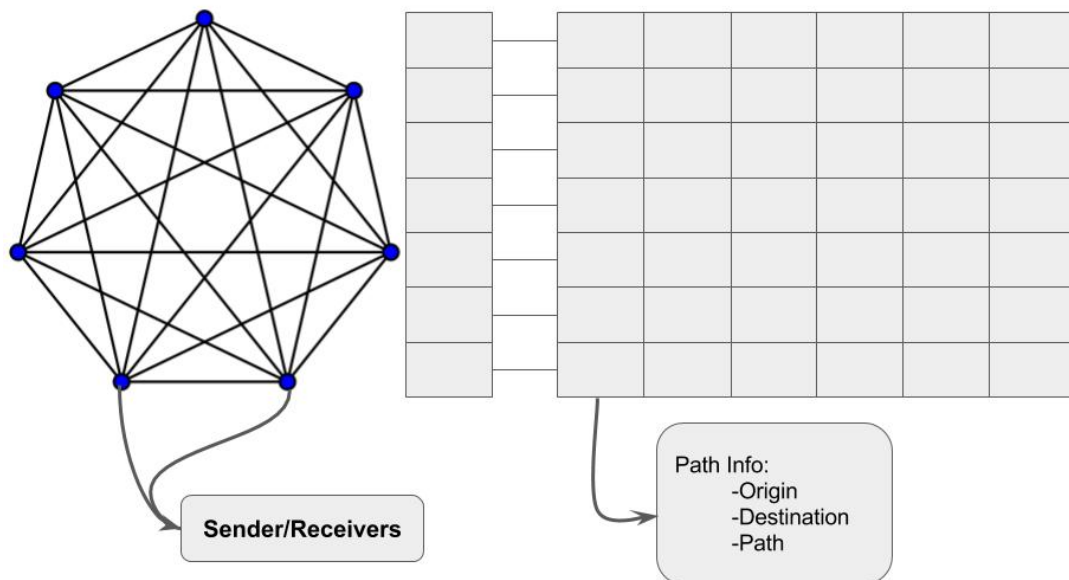


Figure 3.13.: Complete Graph Representation In Parity Maps

Figure 3.13 provides a visual representation to the implemented structure with regards to parity maps. In detail, parity maps represent the complete graph, where a direct connection between each peer is determined, via the Dijkstra algorithm. And so, in each represented cell, there is information regarding the origin, the destination, and the best path between them. This allows for a full-mesh representation between all peers in the session.

Peer Indexation And Cost Association Graph

Having a representation, in memory, of each peer’s shortest path to every other peer in the session, in other words, having a data structure (map) representing the complete graph

(where edges represent the number of hops in the shortest path between peers), allows for the creation of a matrix indexed by peer (Peer x Peer) where each cell contains the cost (number of hops) of travel from source to destination. In Figure 3.14, two examples were included. Overlay nodes 0 and 1 are connected with a shortest path involving 5 ISP links, and overlay nodes 5 and 6 are connected with a shortest path of 2 ISP links.

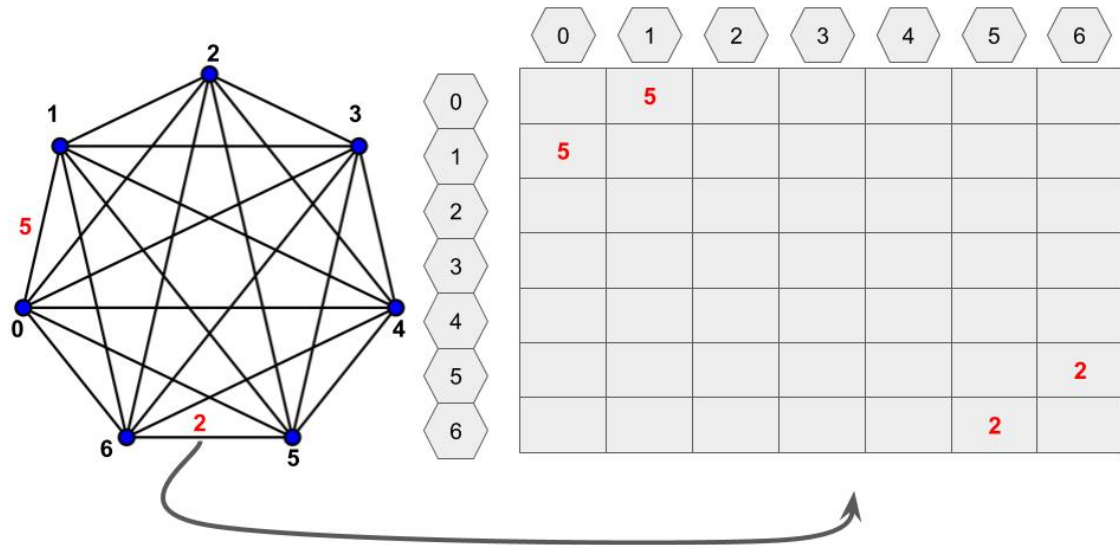


Figure 3.14.: Complete Graph Matrix Representation

Minimum-Spanning-Tree (PRIM)

The work described in the previous section has been performed to get data in proper form so that a Minimum Spanning Tree (MST) may be calculated. An MST is an edges subset, of a connected, weighed and undirected graph (as the demonstrated case), that has no cycles and where all edges are connected with the minimum possible combined edge weights. In other words, the sum of the weights in the graph’s edges is as little as possible. In the presented case, these costs are translated in the number of links connecting two specific peers. With this in mind, after the MST has been calculated, a new multicast tree can be derived, where the number of links being used is the least possible.

Now, there are several algorithms to determine Minimum-Spanning-Trees, the elected one was PRIM (PRIM’s Minimum Spanning Tree [48]), a greedy algorithm that performs as mentioned later in Section 4.1.1.

New Multicast Tree

Taking into consideration that the output from Section 3.3.2 is a vector where the value in each cell represents the parent of the indexed position, the translation of said output into a multicast tree becomes trivial.

So, at this point, the central node has the new overlay multicast tree which minimizes the number of traversed ISP links.

Now, the way operations of this type, where changes are to be made in the connections between peers, have been performed in previous scenarios (such as link protection, for instance), is to have the peers reconnect themselves into the session and, then, manipulate the way they perform their connection procedure. However, in this case, this would not be the most efficient solution as changes will probably be made throughout the whole multicast tree, which would mean flooding both the overlay network and the central node with unnecessary traffic, namely requests of information on the multicast session (requests to the network) and QoS probes (to other peers) which would, again, flood the underlying topology.

With the previous point in mind, a decision was made to create a new kind of message that would be directed to each peer in the session. So, instead of notifying each peer that it must reconnect, the central node now, via a *Link Minimization Tree Update Message*, simply messages each peer its new upstream peer and downstream list. There are no connection processes being performed as they are unnecessary as the central node is now determining connections.

Post Minimization Operating Protocol

An important fact to take into consideration is what happens after the link minimization protocol is in place. Also, in a link minimization scenario, a necessity arises to determine what happens to an arriving peer in a session where the link minimization protocol is in place.

To this point, two decisions were made. When a peer requests the central node information on the multicast session (which typically would result in the central node simply sending it the list of all peers in the session), the central node now:

1. For each peer already in the session, determines shortest path between such peer and the joining peer and elects the result with the least number of hops (the shortest path). It then informs the joining peer of its new upstream peer, and requests the elected peer to update its downstream list with the joining peer;
2. Every N entries of new peers into the session, the Link Minimization protocol is re-performed and the whole multicast session is updated. This approach allows the

reduction of the overlay's oscillations due to the PRIM algorithm outputs, only periodically updating the overlay state.

3.3.3 *ISP Forwarder Activation*

The present section regards yet another form of interaction and cooperation between the central node and the ISP. Once more, the objective is to minimize the number of links receiving the overlay's traffic.

With the goal being similar to the one presented in the Link Minimization section, it is important to understand that the starting point for the present approach, is with sessions implementing the link minimization scenario. The previously mentioned methods/approaches contemplated a somewhat passive interaction with the ISP, it would simply perform requests to the central node (link protection and/or link minimization) and provide it with both the updated network topology and the access router being used by peers in the overlay network, and, with this information, the central node, when possible, would adapt multicast trees to respond to the performed requests. In this new usability context, the ISP takes a much more active role in its implementation.

As mentioned, the starting point for this new approach is a session that already is implementing the link minimization procedures, and it has been shown that sessions implementing these procedures already use as few links as possible, while still ensuring every peer in the session has access to the data being streamed. So, in order to minimize the number of used links even further, it is required that the ISP provides the session (and the central node) with more tools so that it may test if any improvements can be made.

The mentioned tools come in the form of ISP activated forwarders. They are application level entities that the ISP places along strategic places in the network so that the central node can simulate them as peers in the session, and determine if, with the inclusion of these forwarders in the session(s), the number of links in use can be further reduced. In which case, they are then seen as a normal peer in the session, and are assigned an upstream and a downstream list, and, to every other peer in the session, are seen as a "normal" peer.

With the presented generalized view of the approach, let's take a more algorithmic look as to what the performed steps are:

1. The ISP sends the central node a request to activate the link minimization procedures to all multicast sessions;
2. The Central Node, then, activates the appropriate link minimization procedures, as described;

3. The ISP, then, sends the central node a list of its available forwarders in an updated graph of the topology, along with a request that the central node perform the Forwarder Activation Procedures;
4. The central node then, per multicast session:
 - a) For each available forwarder, appends that forwarder to a copy of the session (performed for simulation purposes), and determines the number of ISP links in use;
 - b) Assesses the best scenario and applies the necessary changes, if any;
5. The ISP, should the central node recommend the activation of any certain forwarder, will do as suggested.

With the presented steps in mind, the final aim is to have a, somewhat, pure multicast, where each peer in the session receives the intended data, with the network having been loaded with as little traffic as possible, as a minimum spanning tree is in use.

With the presented steps in mind and with an understanding of what the ultimate goal is, the following will detail the way the process is performed, particularly with regards to step 4 above, as it is where all changes (if any) are determined and applied.

Determining Best Match (Forwarder) and Changes

As stated before, the idea behind this approach, is to place forwarders along the network topology and have them be treated as "normal" peers, included in a multicast session. However, this is to happen only if the inclusion of the forwarder in question results in a lesser number of links being used when distributing data throughout the whole set of peers in each multicast session. Figure 3.15 represents the inclusion of these potential forwarders in the underlying topology (in red), where, data duplication and forwarding may now happen.

However, in order to decide which specific ISP forwarder will be chosen, some simulations must be made so as to determine the best possible scenario before actually implementing any changes.

Figure 3.16 focuses on the best match determination process the central node performs on each multicast session, which includes the following steps:

1. In order to keep the multicast session as stable as possible, tests are not performed directly. This means, copies of the session are performed, as many as the number of potential ISP forwarders;

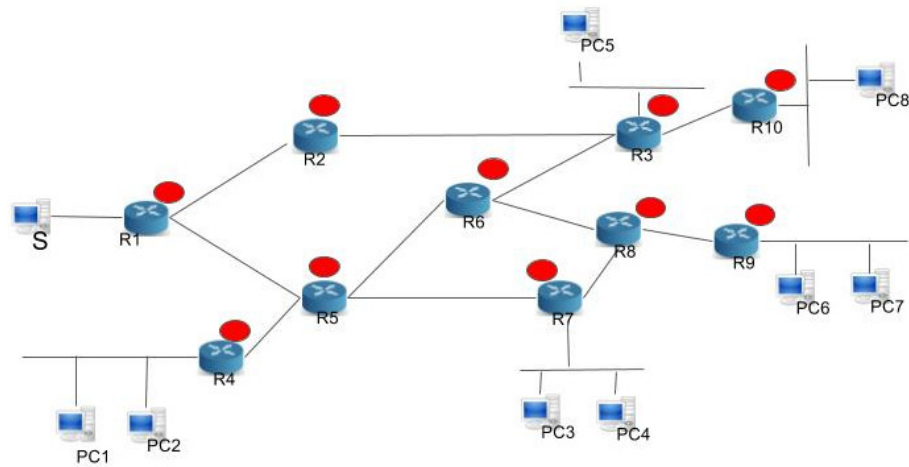


Figure 3.15.: Forwarder Placement

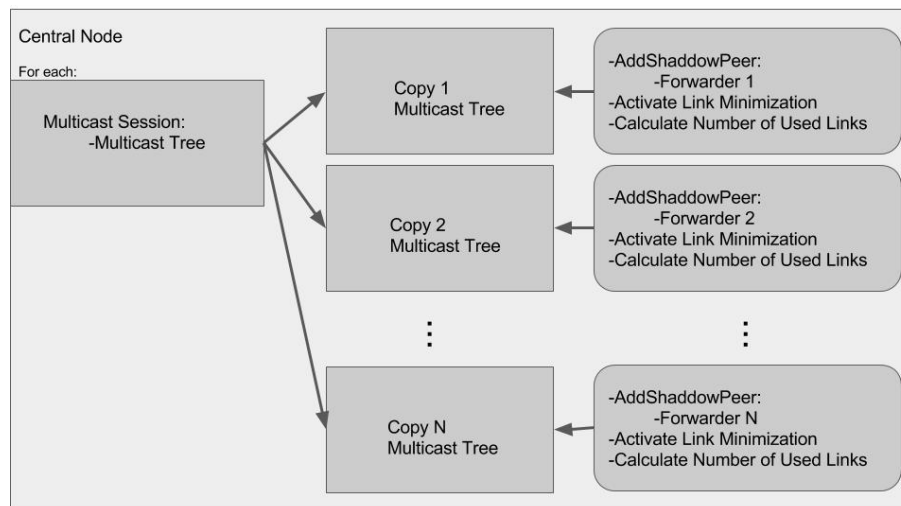


Figure 3.16.: Best Match Determination Logic, N Forwarders

2. In each copy of the session, a *shadow peer* is added, the forwarder being tested;
3. Link Minimization procedures (the same procedures as in 3.3.2) with the newly added peer (forwarder) being recognized as a "normal" receiver in the session;
4. The number of used links is calculated (and stored) using the explained best path determination between each peer connection;

After the described process has been performed, the next step is quite simple, to iterate through the number of different generated multicast trees and determine the one with the least used links, and then, compare it with the number of links in the session already in place:

- Should the number of used links with the specific ISP activated forwarder be lesser than in the session already in place, the forwarder is added to the actual session (not the copy), link minimization procedures are run, and peers are notified of the changes to their upstream peer and downstream list;
- Should the number of used links with forwarder be higher than in the session already in place, all the calculated data is discarded and the session is deemed to be performing better with no changes, so it remains as is.

Figure 3.17 provides a very summarized visual representation on the way the involved entities cooperate in order to make this collaborative approach possible.

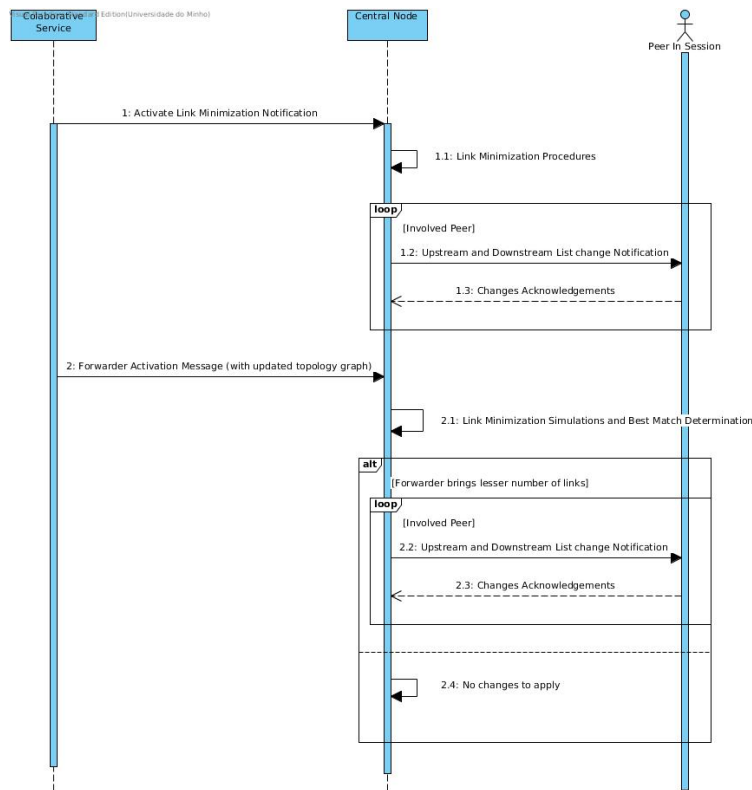


Figure 3.17.: ISP Forwarder Activation Summarized Process

3.4 OVERLAY IN MULTIPLE AUTONOMOUS SYSTEMS

The present section focuses on the extension of the presented work to scenarios contemplating multiple autonomous systems.

The developed architecture and methodologies, as shown thus far and represented in Figure 3.1, show different entities operating in the same autonomous system. However,

an extrapolation can be made to demonstrate that both entities and methodologies can be extended to operate in various autonomous systems, i.e., in much broader scenarios, as demonstrated in Figure 3.18.

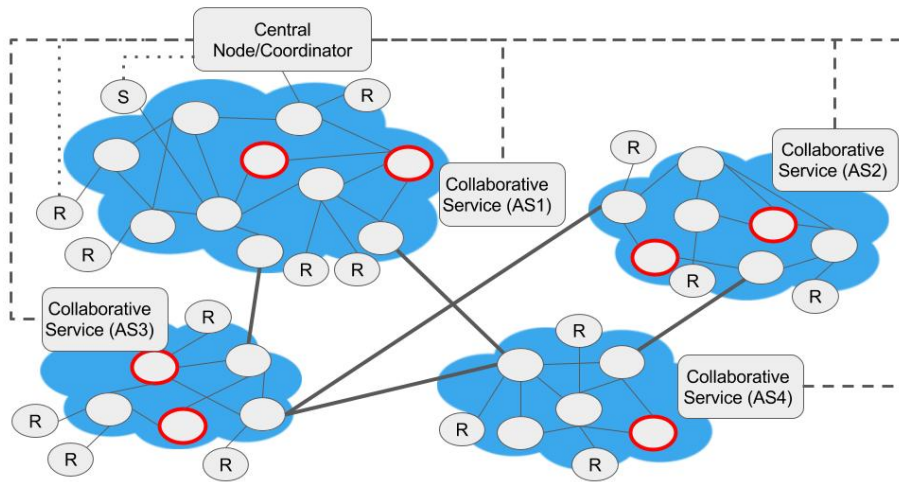


Figure 3.18.: Multiple Autonomous Systems - Expanded Architecture

In Figure 3.18, various autonomous systems are considered, enabling the overlay network to be extended to operate in each of them, as the sender is in AS₁, but the other Autonomous Systems (AS₂, AS₃, AS₄) also contain overlay peers. This results in a wide-spread overlay, distributed and operating through a variety of separate systems.

This extended scenario, however, depends on each Autonomous System having their own collaborative service, in which case, the presented methodologies (such as link protection, minimization, etc) can be applied in a Multi-AS perspective. For these methodologies to be available in such a way, the system only requires that each collaborative service (in a way, representing their own AS) provides the overlay’s central node with its network topology information (much like in the Single-AS scenario) and routes.

Another concept that is introduced with the addition of these systems, is the fact that different collaborative services can eventually communicate among themselves to discuss policies and metrics, and provide the central node with synchronized requests, i.e., requests that they all agree on, such as: *i*) Requesting the central node to try and minimize the number of links the overlay is using; *ii*) Requesting the central node to try to protect a specific set of links, namely, the collaborative services can request that the central node manipulate the way data flows from one Autonomous System to another.

To make the presented approach viable, the only changes required to the system, once more, would be that each AS would have to provide the central node with its topology graphs, each peer’s access router, and its BGP routes. As seen previously, this information

is required so that the central node can, then, simulate the topology and calculate the routing tables for each AS, and apply the necessary changes to the multicast distribution tree.

Assuming the collaborative service (ISP) present in each AS is able and willing to provide the overlay's central node with the required information, the presented symbiotic behavior is possible, enabling the overlay network to enjoy the benevolence and goodwill of the various ASs. On the other end, the different ASs also gain the flexibility of the overlay network, which they can have the central node change and adapt (when possible), in order to better employ their traffic management policies.

TESTING AND RESULTS ANALYSIS

The present chapter will document the tests that validate the implemented strategies and algorithms as well as an analysis on the obtained outcome versus the expected one.

First, a presentation will be made on the technologies used to develop the system, as well as any third party software/libraries, Section 4.1. Then, non-collaborative tests, minimum-delay and minimum-loss, will be performed and documented in Sections 4.2 and 4.3, respectively. Finally, the collaborative approaches will be tested in the following order: Passive and Active Link Protection in Section 4.4, Link Minimization in Section 4.5, ISP Forwarder Activation in Section 4.6, and finally, operation in Multiple Autonomous Systems, in Section 4.7.

4.1 TECHNOLOGIES AND TESTING PLATFORM

To properly develop and test the presented architecture and associated mechanisms, different technologies were required. Various aspects have been taken into consideration in the determination of tools to use, considering the different goals to achieve. The following sections will present the elected tools for the development of the prototype, the Web-Application (and its use), the way the physical network is emulated so tests can be performed and finally, auxiliary libraries that were used to speed the development process and that are instrumental for the system to operate correctly.

4.1.1 *Development Tools*

To accurately implement all the proposed architecture, three separate projects (Peer, Central Node and Collaborative Service) were created, using JAVA 8. The reasoning behind this choice as to do with the numerous advantages of this technology, such as the vast array of available third-party libraries which this work will use, excellent performance, immense documentation, platform ubiquity, etc.

During the prototype implementation, as seen in Sections 3.2 and 3.3, where various methodologies to create and maintain the multicast distribution trees are addressed, the necessity for path determination between any two peers became clear as well as the use of the PRIM algorithm, when link-minimization procedures are put in place. To aid and accelerate this process, two third-party libraries available for JAVA were used, which will now be introduced.

Path Determination

As mentioned before, at any point, the ISP, via the collaborative service, can provide the central node with an updated state of the network topology, information regarding the access router each different peer is using, as well as the required routing information.

The central node, with this information, has to be able to, if possible, reroute overlay traffic through other links in the network. The way this determination on whether or not it is possible to change traffic flow through other links (using the overlay network) is according to both the network topology and the peers that form the multicast session, considering the available routing information.

So, the collaborative service provides the central node with two separate graphs, one containing the network topology and another containing the access point each peer is using into the network. As it will be made clear further in this document, tests for the developed mechanisms will be made in networks using the OSPF (Open Shortest Path First) routing protocol, which uses the Dijkstra algorithm in order to determine the best path between any two source-destination pair.

Now, in order to determine whether or not traffic from a given source to a given destination goes through a specific link, a graph library [47] was used with data structures [49] by the authors, Robert Sedgewick and Kevin Wayne. Figure 4.1 illustrates the generated association of an index array with the nodes that make up the network (which is then inverted in order to facilitate calculations) and, on the right, the number of associations (links) each node (router) has to others.

With this information and representation, it is now possible to determine whether or not traffic between two peers uses any particular link.

PRIM

A second library used in this work has to do with the Link Minimization objective, where the ISP requests that the central node try and form a minimum-spanning tree, having the overlay operate in such a way that the number of ISP links to which the overlay introduces traffic is as small as possible.

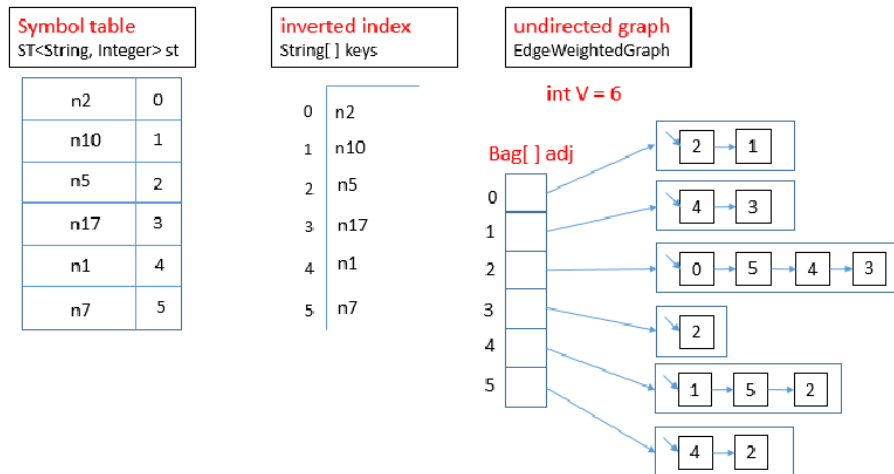


Figure 4.1.: Graph Data Structures

To the previous point, another available library was used [48], which implements, in JAVA, the elected greedy algorithm, PRIM. The following description, logically follows the algorithm:

1. Create a new mstSet, which will store the vertices already included in the MST (changed every step);
2. Initialization:
 - a) Assign a key value to each vertex in the graph;
 - b) Initialize all key values as MAX_INT (Infinite);
 - c) Initialize the source's key value as zero (in this case, the source is the session's sender, so it can be the root of the tree, which is absolutely necessary for best performance);
3. While the mstSet is not complete with vertices in graph:
 - a) Select the vertex with the minimum key value, not considering vertices already in the mstSet pool;
 - b) Include the selected in the mstSet;
 - c) Update the key value for all the vertex's adjacent vertices. For all adjacent vertex:
 - i. if (weight < previous_key_value) key = u-v;

4.1.2 Network Emulation

For testing purposes, the Common Open Research Emulator (CORE), [50] was used to emulate a physical network topology, which is required to test an architecture with the dimension of the proposed one.

This tool provides a very accurate representation of real-life networks, providing scenarios very similar to the ones that the proposed architecture would encounter on deployment. Also, this product is able to simulate not only networks, but also computers, allowing its users to run programs, namely the projects that represent each entity in the proposed architecture, which introduce datagrams into the network. These characteristics, make real simulations to the system possible.

The routing protocol that will be used for the performed tests is OSPF. This protocol implements the Dijkstra algorithm in the determination of the shortest possible path from source until destination. Moreover, with this algorithm, every single node has a global knowledge of the topology for its specific area in the network. In the first scenarios, only one area will be considered. An important consideration to be made is that the CORE network emulator, by omission, attributes a cost of 10 to each link, a cost that will be left unchanged throughout testing.

The testing scenarios that will be presented, will consider the basic network topology presented in Figure 4.2, with any changes to this topology being presented when and where necessary. Any delay or loss in data propagation will be introduced directly into the respective links.

4.1.3 Web-Application

To aid in the analysis of the performed tests, the prototype was complemented with a logging system in order to register requests and decisions each entity makes. Taking into consideration the range the testing scenarios will achieve, it is important to gather the registered logs from each entity in an ordered manner so as to make it possible to see the actions and reactions that occur at each point in the system. So, a web-application was created, which, orders all logs, regardless of the entity, by the time they were made, and so, allows for a step-by-step analysis of the events that occur.

This web application required the use of another set of technologies: *i)* Python & Flask: This technology acts as the server to the web application. It receives requests in order to determine which peers/entities have their logs shown. Obviously, the server is the entity that parses the different files from the various entities and sends the obtained information for rendering; *ii)* HTML 5; *iii)* CSS 3; *iv)* Javascript.

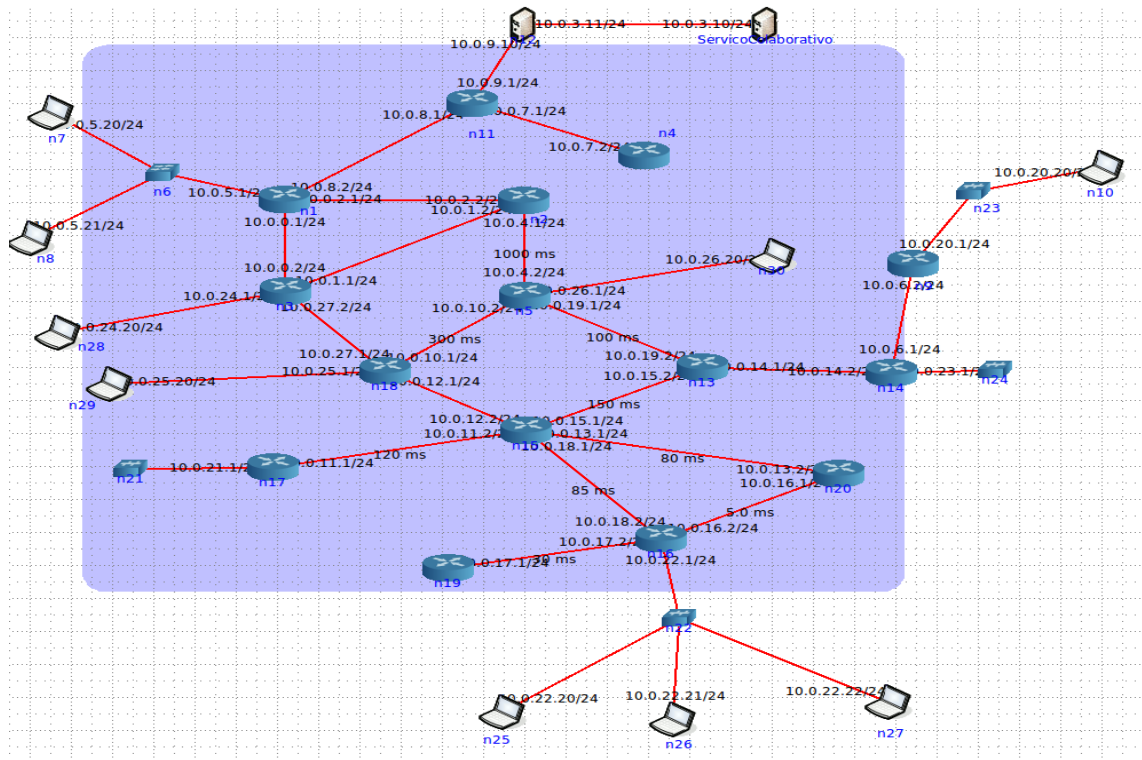


Figure 4.2.: Testing Network Topology

4.2 MINIMUM-DELAY

The first scenario to be tested is one where the usability context's objective is to have the delay between the source and each receiver be the least possible, i.e, the objective, per peer, is to achieve minimum-delay. Table 4.1 considers introduced propagation delay in certain links along the network topology to provide a testing scenario that will enable a better comprehension of the methodology being tested. Other links also have delays, but the ones presented in the mentioned Table aid in the comprehension of the mechanism.

Now, for the purpose of this test, node n8 (10.0.5.21) will act as the sender of the multicast session while each of the other peers will be receivers. So, knowing the algorithm behind the OSPF routing protocol, it is possible to predict the paths that will be used. The paths from source to destination the routing protocol will determine are as follows:

- N8 - N7: Nodes are in the same network;
- N8 - N10: n1(10.0.5.1) - n2(10.0.2.2) - n5(10.0.4.2) - n13(10.0.19.2) - n14(10.0.14.2) - n9(10.0.6.2) - n10(10.0.20.20);
- N8 - (N25, N26, N27) (Network 10.0.22.0/24): n1(10.0.5.1) - n3(10.0.0.2) - n18(10.0.27.1) - n15(10.0.12.2) - n16(10.0.18.2) - 10.0.22.0/24;

Node	Node	Delay (in ms)
n2	n5	1000
n5	n18	300
n5	n13	100
n13	n15	150
n15	n17	120
n15	n16	85
n15	n20	80
n16	n20	5
n16	n19	30

Table 4.1.: Link introduced data delay

- N8 - N28: n1(10.0.5.1) - n3(10.0.0.2) - n28(10.0.24.20);
- N8 - N29: n1(10.0.5.1) - n3(10.0.0.2) - n18(10.0.27.1) - n29(10.0.25.20);
- N8 - N30: n1(10.0.5.1) - n2(10.0.2.2) - n5(10.0.4.2) - n30(10.0.26.29).

This test will be performed according to the following sequence: *i)* Activation of the central node; *ii)* Activation of the collaborative service; *iii)* Activation of peer N8 as the sender; *iv)* Activation of peers as receivers in the following order: N7, N25, N26, N27, N28, N29, N30; *v)* Deactivation of peer N25 (10.0.22.20).

4.2.1 Activating the Central Node

Upon the activation of the central node, two maps are created, one which registers the association between peers and the multicast session they are involved in, and another to maintain the live state of each peer, as the central node employs a soft-state protocol in order to continuously validate that each peer is still in the session, and so, optimize or change the scenario should a given peer leave the session for any reason. With this in mind, three communication ports are activated as the central node starts:

1. Hello Port: this is the port peers send their hello messages to;
2. Management Port: peers use this port to perform every single interaction with the central node other than sending hello messages, meaning, join session requests and notifications, etc;
3. Collaborative Service Port: This port is used to receive the topology graphs as well as any changes to the multicast distribution trees the ISP would like to see happen, mainly network management, as explained.

Figure 4.3 shows the resulting Logs, which are a textual representation of the implemented procedures.

Sequence	Entity	Logger	Message
1	10.0.9.10	10.0.9.10 - Porta Hello	Central Node's Hello Port activated.
2	10.0.9.10	Th_NoCentralPortaGestao	Central Node: Management Port Activated and awaiting.
3	10.0.9.10	10.0.9.10 - Worker Dados Ligacao	WorkerDadosLigacao activated. 5 second interval continuous scanning.
4	10.0.9.10	10.0.9.10 - Porta Hello	Central Node's Hello Port is now listening for messages
5	10.0.9.10	Th_NoCentralPortaServicoColaborativo	Central Node's Collaborative Service Port is Active.
6	10.0.9.10	Th_TrataRecepcaoMensagemServicoColaborativo	Just Received the topology Graph from the Collaborative Service.

Figure 4.3.: Activating the Central Node

4.2.2 Activating the Collaborative Service

Upon activation, the collaborative service immediately sends the central node the graphs with the network topology. The line with sequence number 6, Figure 4.3, represents the central node logging the arrival of the mentioned network topology.

4.2.3 Activating the Sender (Node N8)

As mentioned, the sender of the session will be node N8. Figure 3.7, presented previously in this document, represents the way the peer (sender) and the central node interact in order to create a multicast session. The sender sends a create session message to the central node's management port, the central node verifies that that peer is not involved in any other session and then authorizes the request, registering the session. The sender then assumes the registration has been made on the part of the central node, and starts streaming data. Figure 4.4 represents the registration of events in the creation of the session.

8	10.0.9.10	Th_TrataRecepcaoMensagemPortaGestao	Received a MensagemCriarSessaoMulticast.
9	10.0.9.10	NoCentral	New Multicast Session with the following sender: 10.0.5.21
10	10.0.5.21	Peer	Hello Message Sent.
11	10.0.5.21	Porta Gestão do Peer	Peer's Management Port activated (Th_PeerPortaGestao Notification).
12	10.0.5.21	Porta Gestão do Peer	Type of Peer: Sender
13	10.0.9.10	10.0.9.10 - Porta Hello	Received Hello Message from Peer: 10.0.5.21

Figure 4.4.: Creating a multicast session

4.2.4 Activating a Receiver (Node N7)

Remembering the description of the process peers perform when joining a multicast session is important at this stage:

1. The receiver messages the central node requesting information on the multicast session (namely, the peers involved) via a join session message;
2. The central node replies with the peers involved in the session, should the requested session be a valid one;
3. The receiver checks the central node's reply, then, messages all peers in the session with a QoS (Quality of Service) message, which is time stamped in order to obtain the delay until the source;
4. The receiver, out of the QoS replies, elects a peer to append itself to, and messages that peer in order to be added to its downstream list;
5. The receiver (joining peer) then notifies the central node of said connection.

16	10.0.9.10	Th_TrataRecepcaoMensagemPortaGestao	Received a MensagemJoinSessao.
17	10.0.9.10	NoCentral	No way to protect the Network, sending default list.
18	10.0.9.10	Th_TrataRecepcaoMensagemPortaGestao	Central Node: Sent Multicast Sessions' Peers
19	10.0.9.10	Th_TrataRecepcaoMensagemPortaGestao	Central Node: Sent Multicast Sessions' Peers
20	10.0.5.20	Receiver	List of Possible Connections: 10.0.5.21 -
21	10.0.5.20	Receiver	Received Peers List.

Figure 4.5.: Activating Node 7 - Part 1

Analyzing Figure 4.5, it becomes clear the central node logged the request from the receiver wishing to join the multicast session. As there is no way to protect the network (in this case, because there are not any links to protect at this point), the central node simply replies with every peer in the session. The receiver, through its management port, logs the arrival of the peers list as well as its contents. In this case, there is only one peer to choose from, the sender.

24	10.0.5.21	Th_SenderTrataRecepcaoMensagem	Sender: Received MensagemQoS from Peer: 10.0.5.20
25	10.0.5.21	Th_SenderTrataRecepcaoMensagem	Sender: MensagemQoS Replied.

Figure 4.6.: Activating Node 7 - Part 2

In Figure 4.6, the sender logs the fact that it received a QoS message from the joining peer, logging also the corresponding reply. As node N8 is the sender, its delay until the source is zero ms.

34	10.0.5.20	Th_ReceiverTrataRecepcaoMensagem	Receiver: Received Reply to Join Request.
35	10.0.5.20	Th_ReceiverTrataRecepcaoMensagem	Receiver: Sender accepted my connection, activating Data Port.

Figure 4.7.: Activating Node 7 - Part 3

Figure 4.7 references the joining peer logging the fact that it received a reply from the request it sent, and activated the data port, as the sender has now included peer N7 in its downstream list.

38	10.0.5.20	Th_ReceiverTrataRecepcaoMensagem	Central Node notified of Connection to Sender.
39	10.0.9.10	Th_TrataRecepcaoMensagemPortaGestao	Just Received a MensagemNotificacaoJoinPeer.

Figure 4.8.: Activating Node 7 - Part 4

Figure 4.8 logs the fact that the receiver, node N7, sent the central node a join session notification, informing the central node it had appended itself to the sender, directly in this case. The central node, then, logs this information, and updates its representation of the multicast tree, which can be seen in Figure 4.9, which shows the sender (node N8 - 10.0.5.21) now has one receiver in its downstream list (node N7 - 10.0.5.20), which, itself, has no receivers.

Tree Stage: 1			
Sender		Receivers	Receiver
10.0.5.21	→	10.0.5.20	10.0.5.20

Figure 4.9.: Activating Node 7 - Part 5

4.2.5 Activating other Receivers

As the connection process is analogous for the following receivers, the results of activating peers N25, N26, N27, N28, N29 and N30, will be documented together.

Figure 4.10 (and A.1, in Appendix A.1) provides a visual representation of the multicast tree upon activation of the mentioned peers.

Tree Stage: 7					
Sender		Receivers	Receiver/Transmitter		Receivers
10.0.5.21	→	10.0.5.20 10.0.22.20 10.0.22.21 10.0.22.22 10.0.24.20 10.0.25.20	10.0.25.20	→	10.0.26.20
		Receiver	Receiver		Receiver
		10.0.22.20	10.0.26.20		10.0.22.22
		Receiver	Receiver		
		10.0.22.21	10.0.5.20		

Figure 4.10.: Receivers Activation, Distribution Tree Representation

Figure 4.11 shows the way the peers arranged themselves in order to minimize the delay, where, with a different color, peer N29 is shown forwarding data to peer N30. In other words, the overlay is working on top of the physical network.

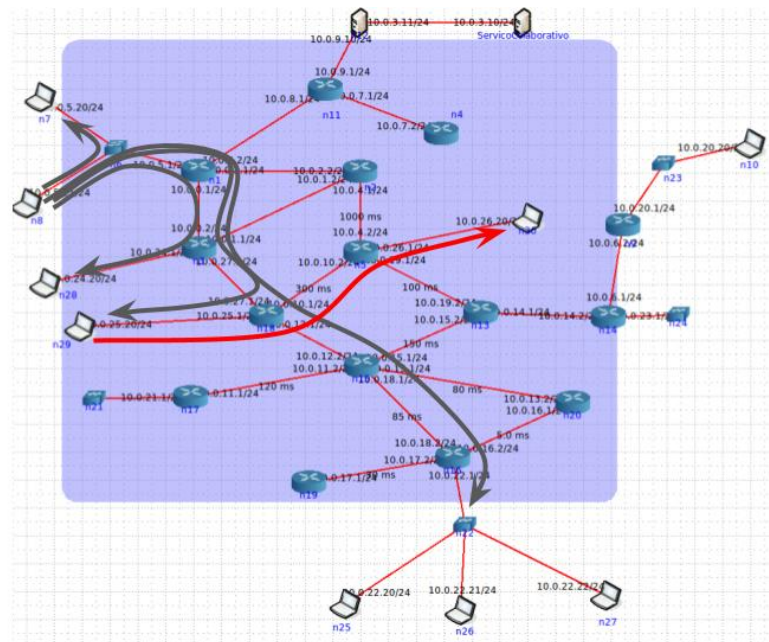


Figure 4.11.: Overlay Distribution Representation

In the paths between the source and each receiver, enforced link delay only occurs twice. Once to the 10.0.22.0/24 network and another in the link between n2 and n5 (through interfaces 10.0.4.1 and 10.0.4.2). Taking a closer look to the sequence of events:

- Nodes in the 10.0.22.0/24 network (nodes N25, N26 and N27) are expected to connect themselves directly to the source, as they share the delay from themselves to the source. Also, as the direct path to peer N7 also includes that delay, it is normal for them to connect themselves directly, instead of adding the extra delay time peer N7 would include;
- Peers N28 and N29 also connect themselves directly to the source, as their respective direct paths to it do not have any delay;
- Peer 30 however, in its direct path to the source, encounters a 1000ms delay, as represented in Figure 4.12. A closer look into the topology, and it becomes clear that the path to peers N28 and N29 also includes a delay, however, it is still faster, and so, peer N30 appends itself to N29, with IP Address 10.0.25.20. The path between peers N8 and N30 is represented in Figure 4.11.

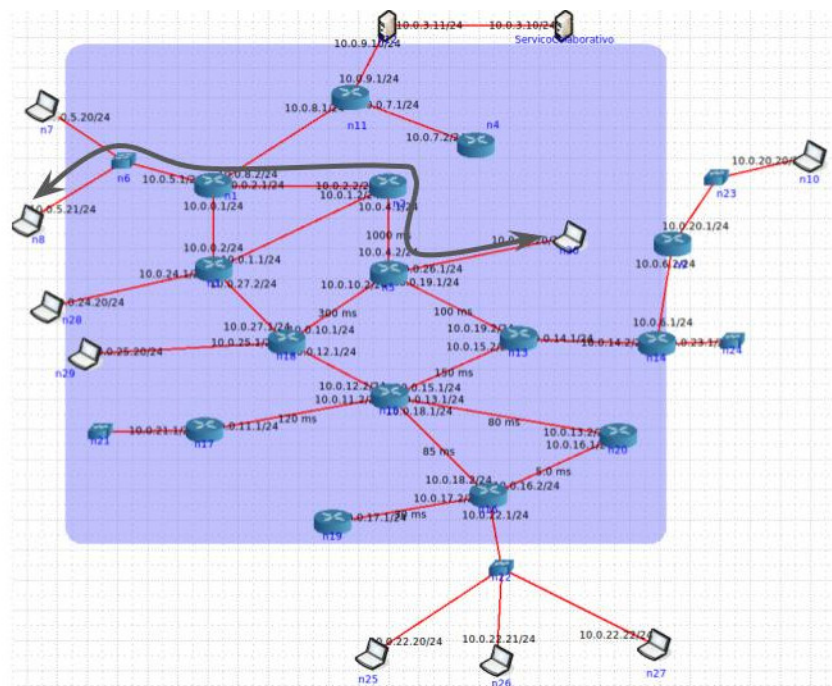


Figure 4.12.: Path between N8 and N30

4.2.6 Deactivating Peer N25 (10.0.22.20)

When a peer leaves the multicast session, for obvious reasons, stops sending hello messages to the central node. As the central node employs a soft-state protocol in order to determine the current state of each peer in each multicast session, within a determined time-space (five seconds), it will quickly be able to determine that a given peer left the session, and will perform the appropriate measures, which are:

1. Determine the upstream river (if any) of the deactivated peer and notify it. This enables the upstream peer to remove it from its downstream list, stopping unnecessary data flow into the network.
2. Determine the children of the deactivated peer (downstream river) and notify each one that they need to establish a new connection with another peer in the multicast session.

Analyzing the current scenario in the deactivation of peer N25, it is clear this peer is neither the source, nor is it forwarding data to any other peer, and so, only its upstream peer needs to be warned to stop sending data to the peer in question. After this removal, the central node updates its multicast tree representation to exclude peer N25. Figure 4.13 contains the generated logs. Figure 4.14 (and Figure A.2 in Appendix A.1) provides a visual representation on the updated multicast tree.

1143	10.0.9.10	10.0.9.10 - Worker Dados Ligacao	Allert: Peer without Hello Message -10.0.22.20;
1144	10.0.9.10	NoCentral	Peer: 10.0.22.20 left. Deploying Measures.
1145	10.0.9.10	NoCentral	Peer Removed from Session.
1146	10.0.9.10	SessaoMulticast	-----Sender: 10.0.5.21 ----- 10.0.25.20 - 10.0.26.20 - 10.0.24.20 - 10.0.26.20 - 10.0.22.22 - 10.0.22.21 - 10.0.5.21 - 10.0.5.20 - 10.0.22.21 - 10.0.22.22 - 10.0.24.20 - 10.0.25.20 - 10.0.5.20 - -----
1147	10.0.5.21	Th_SenderTrataRecepcaoMensagem	Received a MensagemNotificacaoSaidaPeer regarding Peer 10.0.22.20 leaving the Multicast Session. [Removed]

Figure 4.13.: Deactivating Peer N25 Logs

Should peer N25 be forwarding data to any other peers, these would simply be notified as well and would start a new connection establishment process.

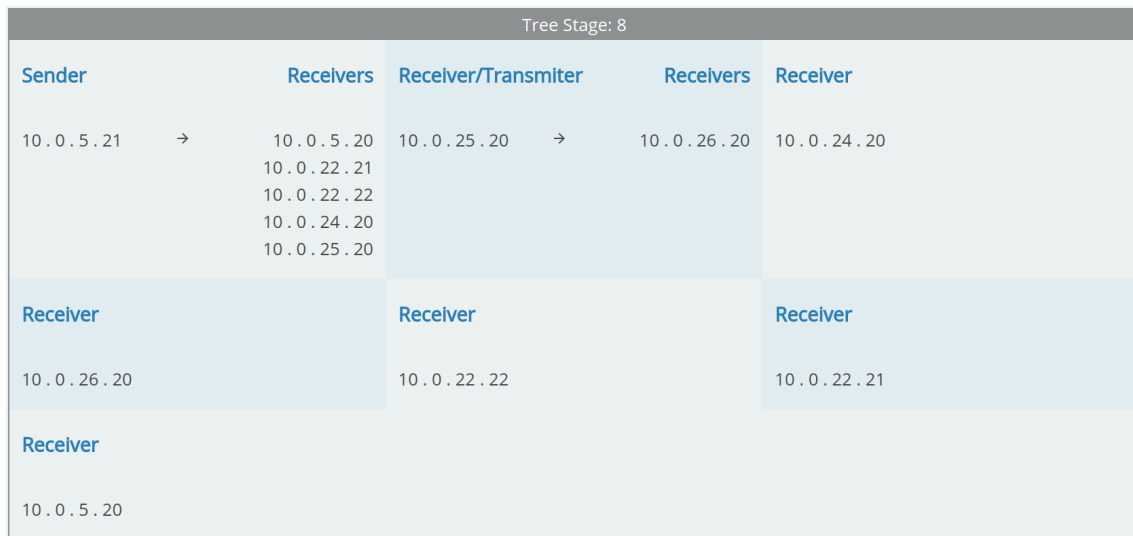


Figure 4.14.: Deactivating Node N25 - Tree Representation 1

Interface	Interface	Packet Loss
10.0.5.20	10.0.5.1	6%
10.0.9.1	10.0.9.10	4%
10.0.0.1	10.0.0.2	12%
10.0.4.1	10.0.4.2	8%

Table 4.2.: Packet Loss assumed in specific links

4.3 MINIMUM-LOSS

The next scenario to be tested contemplates a different usability context, one where the goal is to have any packet loss between sender and receivers be as contained as possible.

Despite the fact that the protocol in use both for multicast sessions and management interactions between central node and peers and, also, between peers themselves, is UDP, the number of lost packets, thus far, has been unnoticeable. This fact, therefore, makes it necessary to force datagram loss in the network being used to test the developed mechanisms. Figure 4.15 presents the updated topology with forced data loss.

Table 4.2 shows packet loss percentages which were introduced directly in the links in question, a functionality insured by the CORE network emulator.

The purpose of the present test is to verify each node selects its appending peer in an appropriate manner, according to the presented selection formula and so, ensuring a maximization of data delivery from source to destination. In order to validate this selection, the following test sequence will be performed: *i)* Activation of the Central Node; *ii)* Activation

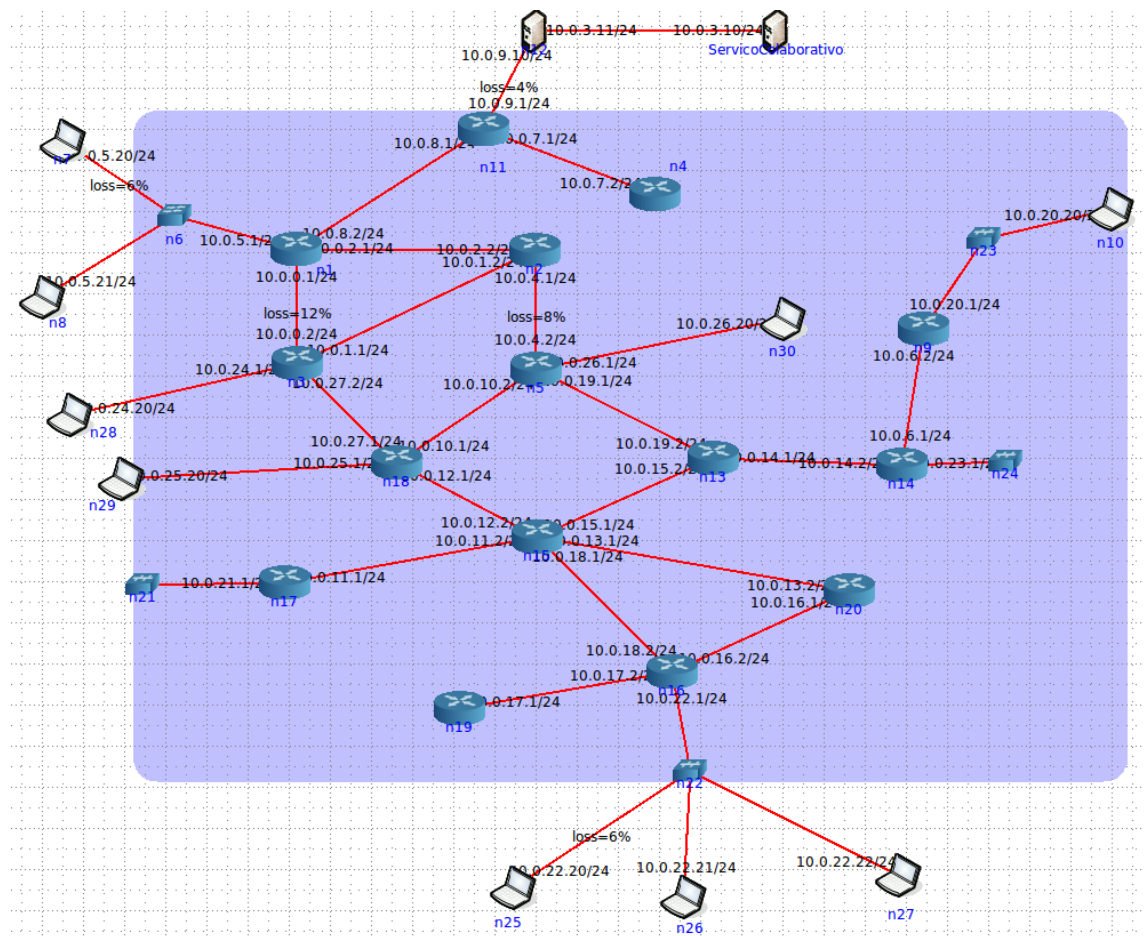


Figure 4.15.: Updated Network Topology

of the Collaborative Service; *iii*) Activation of Peer N8 as the Sender; *iv*) Activation of Peers as receivers in the following order: N7, N28, N30, N29.

4.3.1 Activating the Scenario

As the process until the first receiving peer arrives is the same as before, the scenario activation step will be skipped, which includes steps *i*), *ii*) and *iii*).

Figure 4.16 shows the result of performing the described steps in the presented agenda. Namely:

- The Central Node logging the arrival of the request to create a new minimum-loss multicast session (which is mapped with type "1");
- The Central Node, then, logs the creation of the requested multicast session under the requesting sender, with the requested type;

5	10.0.9.10	Th_NoCentralPortaServicoColaborativo	Central Node's Collaborative Service Port is Active.
6	10.0.9.10	Th_TrataRecepcaoMensagemServicoColaborativo	Just Received the topology Graph from the Collaborative Service.
7	10.0.9.10	Th_TrataRecepcaoMensagemPortaGestao	Received a MensagemCriarSessaoMulticast.
8	10.0.9.10	NoCentral	New Multicast Session with the following sender: 10.0.5.21 and type: 1
9	10.0.5.21	Peer	Hello Message Sent.
10	10.0.5.21	Porta Gestão do Peer	Peer's Management Port activated (Th_PeerPortaGestao Notification).

Figure 4.16.: Activating Scenario Result

- Finally, the peer logs the activation of the necessary ports and its first hello message.

4.3.2 Activating Receiver N7

When activating the first receiver of the session, it is important to remember the described process, in Section 3.2.2, which is summarized in Figure 3.11. So, at this point it is possible to predict the procedure that peer N7 will perform:

- It will message the central node notifying of its intent to join the multicast session where peer N8 (10.0.5.21) is the sender;
- The central node will, then, reply with the IP address of each peer in the session, which, in this particular case, is solely the sender of the session as peer N7 is the first receiver to arrive;
- The QoS metrics will still be performed, despite the fact that the sender is the only peer in the session and, therefore, is the only peer the joining peer can append itself to. This is done so packet loss can still be determined;
- Peer N7 will calculate its packet loss percentage from the sender and perform the connection process to peer N8, then, it will notify the central node, so it may update its version of the multicast tree.

The described procedure is summarized in Figure 4.17 where, in particular, it is clear that peer N7, upon performing the QoS metrics with peer N8, having received 46 out of 50 sent packets, registered a loss percentage of approximately 8%, according to Equation 2. The resulting multicast tree (in its first stage for the presented testing steps) will integrate peer N7 as a receiver of peer N8 (Sender) as predicted.

21	10.0.9.10	Th_TrataRecepcaoMensagemPortaGestao	Central Node: Sent Multicast Sessions' Peers
22	10.0.5.20	Receiver	Received list of peers in the session from the Central Node: 10.0.5.21
23	10.0.5.21	Porta Gestão do Peer	Sender: Received MensagemQoS from Peer: 10.0.5.20, flushing reply (50 packets).
24	10.0.5.21	Peer	Hello Message Sent.
25	10.0.9.10	10.0.9.10 - Porta Hello	Received Hello Message from Peer: 10.0.5.21
26	10.0.9.10	10.0.9.10 - Worker Dados Ligacao	All Peers checked and validated.
27	10.0.5.20	Receiver	Peer: 10.0.5.21: (46/50), with upstream loss percentage 0.0, making total loss percentage: 0.07999998.
28	10.0.5.20	Receiver	Elected Peer 10.0.5.21 making Loss Percentage: 0.07999998
29	10.0.5.21	Porta Gestão do Peer	Sender: Join Message Received.
30	10.0.5.20	Receiver	Received Join Message Reply from Peer 10.0.5.21
31	10.0.5.20	Receiver	Peer 10.0.5.21 accepted my connection Request. Notifying Central Node
32	10.0.9.10	Th_TrataRecepcaoMensagemPortaGestao	Just Received a MensagemNotificacaoJoinPeer.

Figure 4.17.: Logging - Arrival of Peer N7.

4.3.3 Activating Receiver N28

The activation of peer (receiver) N28 follows the same presented logic, despite the single difference that, in this new step, QoS metrics will be performed to more than a single peer, as the session now contemplates two existing peers: N8 and N7.

So, in order to accurately try and "predict" what will happen to the multicast session with the arrival of peer N28, it is important to take a look at Figure 4.15, which represents the topology in place for the presented tests, namely the packet loss percentages in each link, which can also be verified in Table 4.2.

1. Path (N7 - N28):
 - a) 10.0.5.20 - 10.0.5.1: Link with 6% packet loss;
 - b) 10.0.0.1 - 10.0.0.2: Link with 12% packet loss;
 - c) 10.0.24.1 -10.0.24.20: Link with no forced packet loss;
2. Path (N8 - N28):
 - a) 10.0.5.21 - 10.0.5.1: Link with no forced packet loss;
 - b) 10.0.0.1 - 10.0.0.2: Link with 12% packet loss;
 - c) 10.0.24.1 -10.0.24.20: Link with no forced packet loss

Having analyzed the paths between each peer in the session to peer N28, it is expected that peer N28 appends itself to peer N8, as it will probably present a lower packet loss percentage.

99	10.0.24.20	Receiver	Peer: 10.0.5.21: (43/50), with upstream loss percentage 0.0, making total loss percentage: 0.13999999.
100	10.0.24.20	Receiver	Peer: 10.0.5.20: (45/50), with upstream loss percentage 0.07999998, making total loss percentage: 0.17199999.
101	10.0.24.20	Receiver	Elected Peer 10.0.5.21 making Loss Percentage: 0.13999999
102	10.0.5.21	Porta Gestão do Peer	Sender: Join Message Received.
103	10.0.24.20	Receiver	Received Join Message Reply from Peer 10.0.5.21
104	10.0.24.20	Receiver	Peer 10.0.5.21 accepted my connection Request. Notifying Central Node
105	10.0.9.10	Th_TrataRecepcaoMensagemPortaGestao	Just Received a MensagemNotificacaoJoinPeer.

Figure 4.18.: Logging - Arrival of Peer N28.

Figure 4.18 shows Peer N28 logging the fact that it elected to append itself to peer N8, as expected. The remaining logs have to do with the connection establishment process as well as central node notification. It also shows the calculated packet loss percentage between peers N7 and N28.

Figure 4.19 shows the updated multicast tree at the current stage of the testing process.

Tree Stage: 2			
Sender	Receivers	Receiver	Receiver
10.0.5.21	→	10.0.5.20 10.0.24.20	10.0.5.20

Figure 4.19.: Minimum-Loss Tests: Multicast Tree Stage 2.

4.3.4 Activating Receiver N30

With the understanding of the process peers N7 and N28 performed in order to determine the peer they would append themselves to, the decision-making process receiver N30 iterates through becomes clear, as it follows the same logic.

It will test packet loss percentage from each peer in the session, and then elect the peer which results in a lesser cumulative loss percentage from the sender. Looking, once more, at Figure 4.15, and taking into consideration the packet loss percentages in each link, an expected prediction can be made on the peer that will be elected.

1. Path (N7 - N30):
 - a) 10.0.5.20 - 10.0.5.1: Link with 6% packet loss;
 - b) 10.0.2.1 - 10.0.2.2: Link with no forced packet loss;
 - c) 10.0.4.1 - 10.0.4.2: Link with 8% packet loss;
 - d) 10.0.26.1 - 10.0.26.20: Link with no forced packet loss;

2. Path (N8 - N30):

- a) 10.0.5.21 - 10.0.5.1: Link with no packet loss;
- b) 10.0.2.1 - 10.0.2.2: Link with no forced packet loss;
- c) 10.0.4.1 - 10.0.4.2: Link with 8% packet loss;
- d) 10.0.26.1 - 10.0.26.20: Link with no forced packet loss;

3. Path (N28 - N30):

- a) 10.0.24.20 - 10.0.24.1: Link with no forced packet loss;
- b) 10.0.27.2 - 10.0.27.1: Link with no forced packet loss;
- c) 10.0.10.1 - 10.0.10.2: Link with no forced packet loss;
- d) 10.0.26.1 - 10.0.26.20: Link with no forced packet loss;

Now, if a direct connection logic were in place, the expectation would be that the elected peer would be peer N28, as there is no forced packet loss between the joining peer N28 and peer N30. However, it makes sense to consider not only each peer's packet loss percentage from its upstream but also such peer's loss from the source, as a cumulative scenario was implemented. And, as seen before, N28 appended itself to the source with approximately 14% (link with 12% packet loss percentage is used) packet loss percentage from the source. With this in mind, it becomes clear that the path from peers N8 and N7 has less forced packet loss, making peer N8 the expected elected peer.

215	10.0.26.20	Receiver	Peer: 10.0.24.20: (50/50), with upstream loss percentage 0.13999999, making total loss percentage: 0.13999999.
216	10.0.26.20	Receiver	Peer: 10.0.5.21: (46/50), with upstream loss percentage 0.0, making total loss percentage: 0.07999998.
217	10.0.26.20	Receiver	Peer: 10.0.5.20: (48/50), with upstream loss percentage 0.07999998, making total loss percentage: 0.11680001.
218	10.0.26.20	Receiver	Elected Peer 10.0.5.21 making Loss Percentage: 0.07999998
219	10.0.5.21	Porta Gestão do Peer	Sender: Join Message Received.
220	10.0.26.20	Receiver	Received Join Message Reply from Peer 10.0.5.21
221	10.0.26.20	Receiver	Peer 10.0.5.21 accepted my connection Request. Notifying Central Node
222	10.0.9.10	Th_TrataRecepcaoMensagemPortaGestao	Just Received a MensagemNotificacaoJoinPeer.

Figure 4.20.: Logging - Arrival of Peer N30.

Figure 4.20 shows the generated Logs with regards to Peer N30's arrival, and election of peer N8 as its upstream, as expected. It shows the calculated packet loss percentage between peers N28 and N30, which, as explained, is peer N28's packet loss percentage from the source, as no packet was lost in the QoS test process between these two peers. It also considers the packet loss percentage from peer N7 to the joining peer, N30.



Figure 4.21.: Minimum-Loss Tests: Multicast Tree Stage 3.

Figure 4.21 shows the updated multicast tree at the current stage of the testing process.

4.3.5 Activating Receiver N29

The last part of the proposed test, regards the activation of receiver N29. Once more, the election process will be the same as previously.

Analyzing the network topology, it becomes clear that direct connections with peers N8 and N7 involve direct packet loss, whereas connections to peers N28 and N30 would involve the upstream’s packet loss percentages. Well, direct connections to N8 and N7 involve two links with (rather high) loss percentages, which would also be involved in a connection with peer N28. On the other side, a connection with peer N30, would involve, solely, one link with packet loss percentage from the source, and a lesser one. And so, the expected elected peer is N30. Figure 4.22 shows peer N29’s decision making, calculating the respective packet loss percentages, per peer, and electing peer N30 as its source.

337	10.0.25.20	Receiver	Peer: 10.0.24.20: (50/50), with upstream loss percentage 0.13999999, making total loss percentage: 0.13999999.
338	10.0.25.20	Receiver	Peer: 10.0.26.20: (50/50), with upstream loss percentage 0.07999998, making total loss percentage: 0.07999998.
339	10.0.25.20	Receiver	Peer: 10.0.5.21: (42/50), with upstream loss percentage 0.0, making total loss percentage: 0.16000003.
340	10.0.25.20	Receiver	Peer: 10.0.5.20: (42/50), with upstream loss percentage 0.07999998, making total loss percentage: 0.22720003.
341	10.0.25.20	Receiver	Elected Peer 10.0.26.20 making Loss Percentage: 0.07999998
342	10.0.26.20	Porta Gestão do Peer	Just Received a MensagemJoinPeer.
343	10.0.25.20	Receiver	Received Join Message Reply from Peer 10.0.26.20
344	10.0.25.20	Receiver	Peer 10.0.26.20 accepted my connection Request. Notifying Central Node
345	10.0.9.10	Th_TrataRecepcaoMensagemPortaGestao	Just Received a MensagemNotificacaoJoinPeer.

Figure 4.22.: Logging - Arrival of Peer N29.

Figure 4.23 shows the final version of the multicast tree, with peer N29 having elected peer N30 (now shown not only as a receiver but also as a transmitter) as its upstream source, with a packet loss percentage equivalent to the one peer N30 has from the session’s source.

Tree Stage: 4				
Sender	Receivers	Receiver/Transmitter	Receivers	Receiver
10.0.5.21 →	10.0.5.20 10.0.24.20 10.0.26.20	10.0.26.20 →	10.0.25.20	10.0.25.20
Receiver		Receiver		
10.0.24.20		10.0.5.20		

Figure 4.23.: Minimum-Loss Tests: Multicast Tree Stage 4.

4.4 PROTECT-LINK

In these tests, the same topology was used as in Section 4.2, with the same testing conditions, meaning, link delays are unchanged for this scenario. Furthermore, this section represents a system protection that is independent of the multicast tree’s usability context, it has to work regardless of the metric that is being used in the formation of the multicast tree. For this test, the tree will maintain a minimum-delay construction scenario.

4.4.1 Passive Protection

To test the developed algorithm for passive network protection, with the overlay operating under the minimum-delay usability context, similar overlay multicast trees are expected as the ones in Section 4.2. However, even though peers will use the same metric, the performed steps will be somewhat different:

1. Activation of the central node;
2. Activation of the collaborative service;
 - Collaborative service protects link N18-N15;
3. Activation of peer N8 as the sender;
4. Activation of Peers N7, N25, N26 and N27 as receivers.

Although the presented test scenario provides for a smaller test case, passive protection is expected, as the direct path from peers in the 10.0.22.0/24 network to the peers N7 and N8 includes the link to protect.

Activation of the Central Node

The activation of the central node, as expected, follows the same steps of the activation in the previous example, 4.2. Figure 4.24 represents the resulting logs.

Sequence	Entity	Logger	Message
1	10.0.9.10	10.0.9.10 - Porta Hello	Central Node's Hello Port activated.
2	10.0.9.10	10.0.9.10 - Porta Hello	Central Node's Hello Port is now listening for messages
3	10.0.9.10	10.0.9.10 - Worker Dados Ligacao	WorkerDadosLigacao activated. 5 second interval continuous scanning.
4	10.0.9.10	Th_NoCentralPortaGestao	Central Node: Management Port Activated and awaiting.
5	10.0.9.10	Th_NoCentralPortaServicoColaborativo	Central Node's Collaborative Service Port is Active.

Figure 4.24.: Activating the Central Node

Activation of the Collaborative Service and N15-N18 Link Protection

Once more, upon activation, the collaborative service immediately sends the central node the graphs with the network topology representation, Figure 4.25 shows the central node logging the arrival of the topology graphs.

The collaborative service, having been activated and having sent the network topology, then, sends a message to the central node, requesting the protection of link N15-N18. Figure 4.26 shows how the ISP uses the collaborative service to interact with the central node. In this case, requesting it protect the link in question.

8	10.0.9.10	Th_TrataRecepcaoMensagemServicoColaborativo	Just Received the topology Graph from the Collaborative Service.
9	10.0.9.10	10.0.9.10 - Worker Dados Ligacao	All Peers checked and validated.
10	10.0.9.10	10.0.9.10 - Worker Dados Ligacao	All Peers checked and validated.
11	10.0.9.10	Th_TrataRecepcaoMensagemServicoColaborativo	Just Received a Request to protect Link (n18 - n15) from the Collaborative Service.

Figure 4.25.: Activating the collaborative service - Part 1

```

1 - Re-Send Topology Graph
2 - Protect Link
3 - Remove Link Protection
Option: 2
Origin Node: n18
Destination Node: n15

```

Figure 4.26.: Activating the collaborative service - Part 2

Activating the Sender (Peer N8)

As with the previous scenario, the sender for this multicast session will be node N8. Again, the steps in creating a new session are the same as before, Figure 4.27 shows the resulting Logs, both by the central node and the sender.

16	10.0.9.10	Th_TrataRecepcaoMensagemPortaGestao	Received a MensagemCriarSessaoMulticast.
17	10.0.9.10	NoCentral	New Multicast Session with the following sender: 10.0.5.21
18	10.0.5.21	Peer	Hello Message Sent.
19	10.0.5.21	Porta Gestão do Peer	Peer's Management Port activated (Th_PeerPortaGestao Notification).
20	10.0.5.21	Porta Gestão do Peer	Type of Peer: Sender
21	10.0.9.10	10.0.9.10 - Porta Hello	Received Hello Message from Peer: 10.0.5.21

Figure 4.27.: Logs regarding the activation of the sessions' sender

Activating Receiver - Peer N7

Looking at the topology represented in Figure 4.2 and considering the OSPF routing algorithm, it is clear the path from peer N7 and the sender does not include the link to protect. And so, the connection is established normally. Figure 4.28 represents the logs resulting from peer N7's activation and Figure 4.29 shows the visual representation of the resulting multicast tree.

29	10.0.9.10	Th_TrataRecepcaoMensagemPortaGestao	Received a MensagemJoinSessao.
30	10.0.9.10	Th_TrataRecepcaoMensagemPortaGestao	Central Node: Sent Multicast Sessions' Peers
31	10.0.9.10	Th_TrataRecepcaoMensagemPortaGestao	Central Node: Sent Multicast Sessions' Peers
32	10.0.5.20	Receiver	List of Possible Connections: 10.0.5.21 -
33	10.0.5.20	Receiver	Received Peers List.

Figure 4.28.: Logs regarding the activation of peer N7

Tree Stage: 1			
Sender		Receivers	Receiver
10.0.5.21	→	10.0.5.20	10.0.5.20

Figure 4.29.: Multicast Tree Stage 1

Activating Receiver - Peer N25

The path between peer N25 and both other peers in the session includes the link to protect.

Now, it has been stated before that, both passive and active protection modes will protect the network, if possible. This is a case where it is not possible to protect the network, as the resulting paths from N25 to N8 and N7, with the OSPF routing protocol, include the link to protect, as represented in Figure 4.30. So, the central node has no possible solution, other than allowing for the connection of peer N25 to any peer in the session, and so, not being able to protect the network. With the presented reasons in mind, the normal connection establishment process is performed.

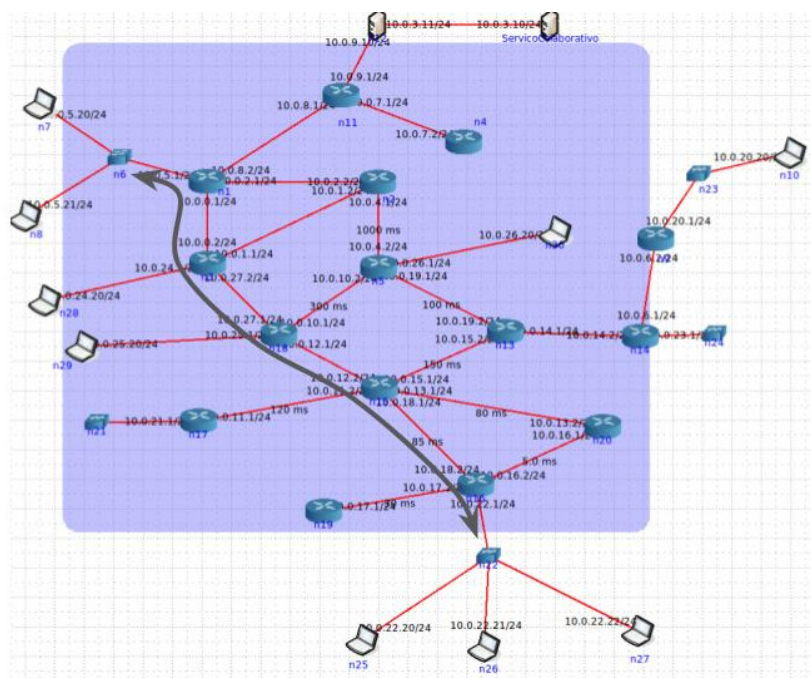


Figure 4.30.: Path from peer N25 to peers N7 and N8

As the result of a standard connection process, node N25 elected to append itself directly to the source of the multicast session, as seen in Figure 4.31. Figure 4.32 provides a visual representation of the overlay network.

Tree Stage: 2			
Sender	Receivers	Receiver	Receiver
10.0.5.21	→	10.0.5.20 10.0.22.20	10.0.22.20 10.0.5.20

Figure 4.31.: Multicast Tree stage 2 - Representation

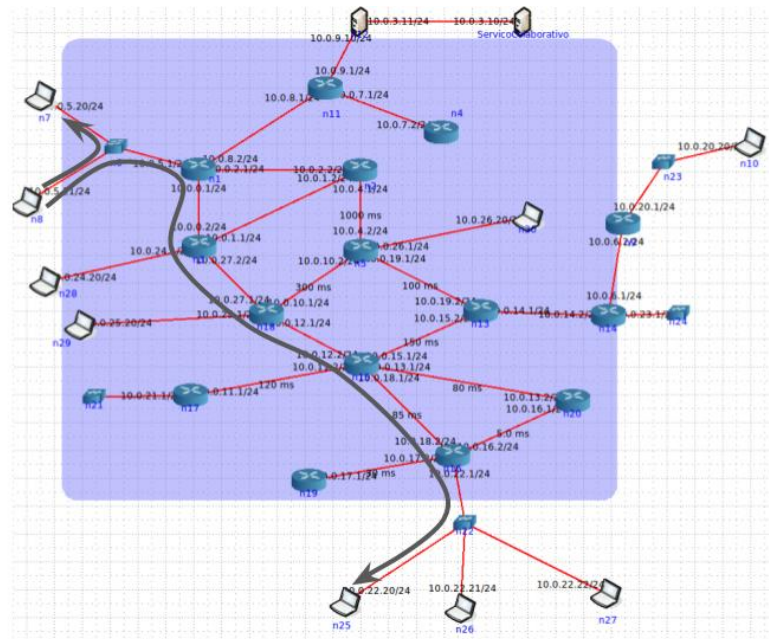


Figure 4.32.: Passive Protection: Overlay Distribution Tree

Activating Receivers - Peers N26 and N27

Now, peers N26 and N27 belong to the same network as peer N25, network 10.0.22.0/24. Obviously, looking at the topology once more, they share the same path until peers N7 and N8. It has also been stated that it is impossible to protect data flow into the link N15-N18, which is the link to protect. However, it is possible to have traffic through that path only once, and having peer N26 connect itself directly to N25 and then, N27 choose from the two (likely connecting itself to peer N25 as well due to the forwarding added time).

166	10.0.22.21	Receiver	List of Possible Connections: 10.0.22.20 -
-----	------------	----------	--

Figure 4.33.: Central Node's reply to peer N26

So, whilst data streams from the source to peer N25, peer N26, when contacting the central node for the list of peers it can append itself to, receives only one possible connection in the central node’s reply, as seen in Figure 4.33.

Upon activation of peer N27, the central node repeats the same process, and peer N27 connects itself with the same logic. Figure 4.34 (and Figure A.3 in Appendix A.2) provides a visual representation on the final multicast tree obtained for this test, making it possible to conclude that the prototype worked as expected. Showing peer N25 connected directly to the source, and peers N26 and N27 being forwarded data by peer N25. Another representation is provided in Figure 4.35, where data paths and peer connections are made clear. So, even though it is impossible to completely protect the link in question, the data flowing through it can, at least, be reduced.

Tree Stage: 4					
Sender		Receivers	Receiver/Transmitter		Receivers
10.0.5.21	→	10.0.5.20 10.0.22.20	10.0.22.20	→	10.0.22.21 10.0.22.22
Receiver			Receiver		
10.0.22.21			10.0.5.20		

Figure 4.34.: Multicast Tree Stage 4

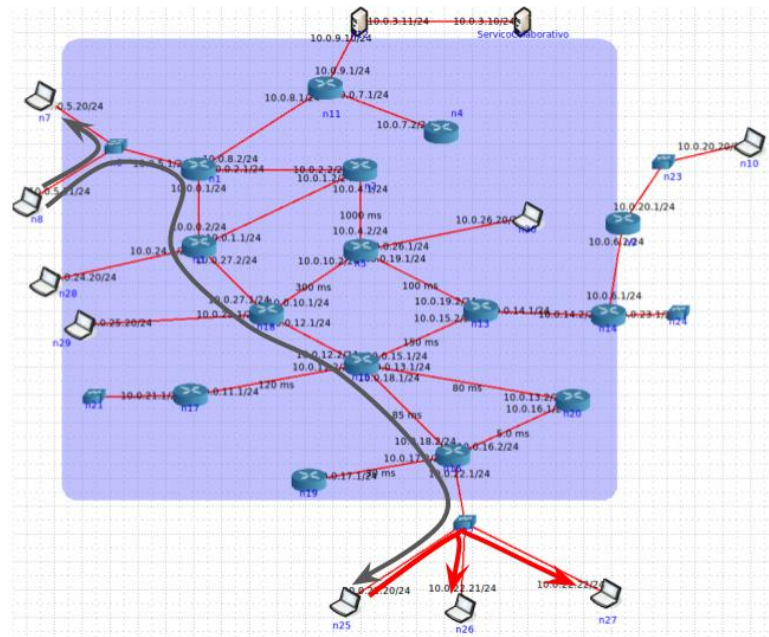


Figure 4.35.: Passive Protection: Final Overlay Distribution Tree

4.4.2 Active Protection

Active protection, as described, is a more aggressive method of protecting the network. Not only does it mean that new connections are to avoid the passage of data through the link to protect, but also, existing connections are to be interrupted, and the receiving peers to append themselves to other peers in the network, peers whose path does not include the protected link.

With the described capacity in mind, it is important to describe the test scenario. The following steps will be taken: *i)* Activation of the central node; *ii)* Activation of the collaborative service; *iii)* Activation of Peer N8 as the sender; *iv)* Activation of Peers N7, N25, N26, N27, N28, N29 and N30 as receivers; *v)* Collaborative Service requests the protection of link N5-N18.

As it is possible to see from the steps to be taken, and considering, once more, the usability context that will be used, in this case is one with a minimum-delay for each peer, in order to go directly to the active protection system, the scenario will be activated completely until the protection step.

Activating Scenario

With the presented usability context, the multicast tree, before the collaborative service requests the protection of link N5-N18, must be similar to the one seen in section 4.2, as expected, until the deactivation of certain peers. Figure 4.36 provides the the visual representation on the central node's map of the multicast tree whereas Figure 4.37 shows the overlay tree formed on top of the physical network.

Protecting Link N5-N18

Knowing the link to protect is the one connecting routers N5 and N18, the problem becomes quite clear. As link N5-N18 is to be protected, this will affect the connection between peers N29 and N30, as peer N29 is forwarding data to peer N30, with the path highlighted in Figure 4.38. So, at this point, when the ISP uses the collaborative service to request that link N5-N18 be protected:

1. The central node must add this link to the protection list;
2. The central node must determine whether or not there are any two peers (or more) passing data through the link in question. Should this be the case:

Tree Stage: 7					
Sender	Receivers	Receiver/Transmitter	Receivers	Receiver	
10.0.5.21 →	10.0.5.20 10.0.22.20 10.0.22.21 10.0.22.22 10.0.24.20 10.0.25.20	10.0.25.20 →	10.0.26.20	10.0.24.20	
Receiver		Receiver		Receiver	
10.0.22.20		10.0.26.20		10.0.22.22	
Receiver		Receiver			
10.0.22.21		10.0.5.20			

Figure 4.36.: Tree before protection (Stage 7)

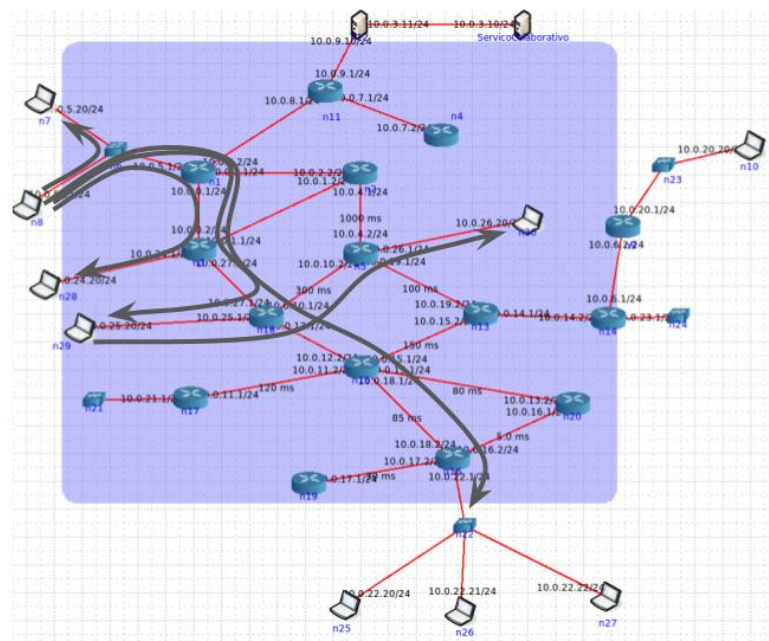


Figure 4.37.: Active Protection: Overlay Distribution Tree

- a) The peer forwarding data is to be notified to remove the receiving peer from it's downstream list;
- b) The receiving peer is to be notified to establish a new connection to another peer, with the protection system falling back to passive protective mode, as the new connecting peer will now try to enter the multicast session again with link protection in place.

- c) No other peers in the session should even be aware of the changes in the network and multicast tree;
3. The receiving peer then starts the normal connection procedures.

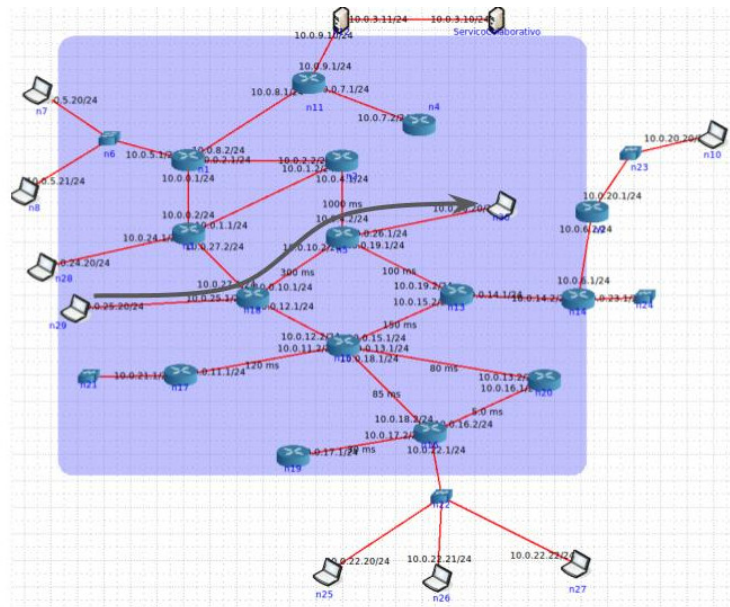


Figure 4.38.: Path between peers N29 and N30

Figure 4.39 represents the collaborative service’s request to the central node that link N5-n18 be protected.

```

1 - Re-Send Topology Graph
2 - Protect Link
3 - Remove Link Protection
Option: 2
Origin Node: n5
Destination Node: n18
    
```

Figure 4.39.: Collaborative Service’s request to protect link N5-N18

Figure 4.40 shows the central node’s log regarding the fact that it received the collaborative service’s request to protect link N5-N18 and will deploy the appropriate measures, which are to identify the problematic connections, meaning peer pairings using the link in question. Figure 4.41 shows the result of the central node’s queries regarding the multicast tree. So, the central node has determined, as expected, that there is an established connection between peers N29 and N30, where peer N29 (called parent) is forwarding data to peer N30 (which is stamped as problematic).

```
INFO: Just Received a Request to protect Link (n5 - n18) from the Collaborative Service.
```

Figure 4.40.: Central Node Logs Receiving the collaborative service's request

```
Parent: 10.0.25.20
Problematic: 10.0.26.20
```

Figure 4.41.: Central node identification of problematic peers

For this specific scenario, having determined the two peers with an established connection using a path with a link to protect, the central node, then, simply has to notify peer N29 to remove peer N30. Also, the central node, must have peer N30 re-establish its connection to the multicast session. Figure 4.42 represents Peer N29 logging the fact that it removed peer N30 from its downstream list.

991	10.0.25.20	Th_ReceiverTrataRecepcaoMensagem	Removed Peer 10.0.26.20 from downstream list.
-----	------------	----------------------------------	---

Figure 4.42.: Peer N29 logging the removal of peer N30 from downstream list

Figure 4.43 represents the fact that peer N30 got an order to re-establish its connection to the multicast session, and also, the fact that, when the central node sent the list of possible peers N30 could connect itself to, peer N29 (as well as N28, which would use the same link) is not included in the list, and so, the network is protected, as the paths to the possible peers in the central node's reply are represented in Figure 4.44.

```
Mar 26, 2017 4:10:16 PM Peer.Receiver setupLigacao
INFO: Received Peers List.
1 - 10.0.22.20
1 - 10.0.22.22
1 - 10.0.22.21
1 - 10.0.5.21
1 - 10.0.5.20
-----
Mar 26, 2017 4:10:19 PM Peer.Peer enviarMensagemHello
INFO: Hello Message Sent.
Now checking MensagemQoS results, obtained answers: 5
-----List Qos Messages -----
1 - Reply from Peer 10.0.22.22 with following delay: 121
2 - Reply from Peer 10.0.22.21 with following delay: 114
3 - Reply from Peer 10.0.22.20 with following delay: 125
4 - Reply from Peer 10.0.5.21 with following delay: 0
5 - Reply from Peer 10.0.5.20 with following delay: 45
-----
```

Figure 4.43.: Peer N30 is requested to establish new connection

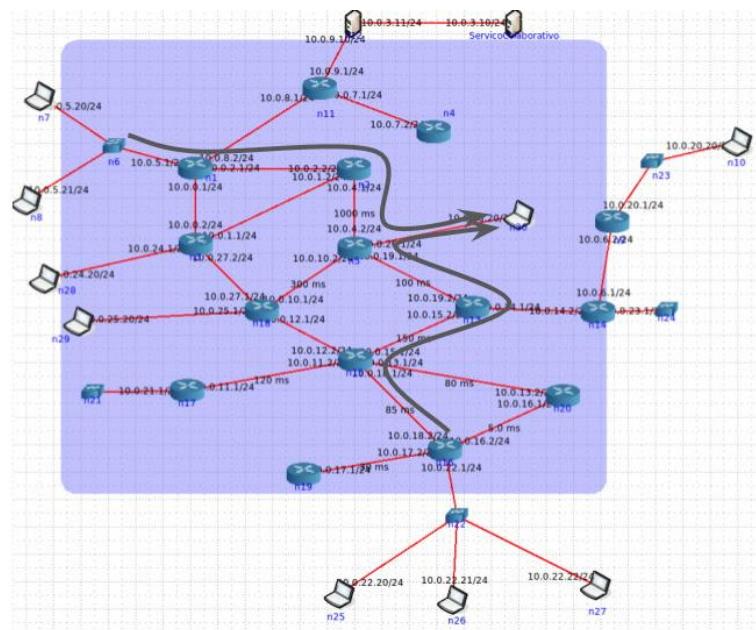


Figure 4.44.: Peer N30, paths to possible connections

As the normal connection scenario is now being used, due to the fact that peers N28 and N29 are not on the list of possible connections, the network has been protected, as expected. And so, finally, the representation of the final stage multicast tree is presented in Figure 4.45, showing peer N30 being forwarded data by peer N27, and so avoiding the one second delay it would have with peers N7 and N8. Figure 4.46 shows the overlay network, once more, working on top of the physical network, in order to protect the mentioned link.

Tree Stage: 8					
Sender	Receivers	Receiver/Transmitter	Receivers	Receiver	
10.0.5.21	→	10.0.5.20 10.0.22.20 10.0.22.21 10.0.22.22 10.0.24.20 10.0.25.20	10.0.22.22	→	10.0.26.20
	Receiver		Receiver		Receiver
	10.0.24.20		10.0.22.20		10.0.26.20
	Receiver		Receiver		
	10.0.22.21		10.0.5.20		

Figure 4.45.: Final multicast tree

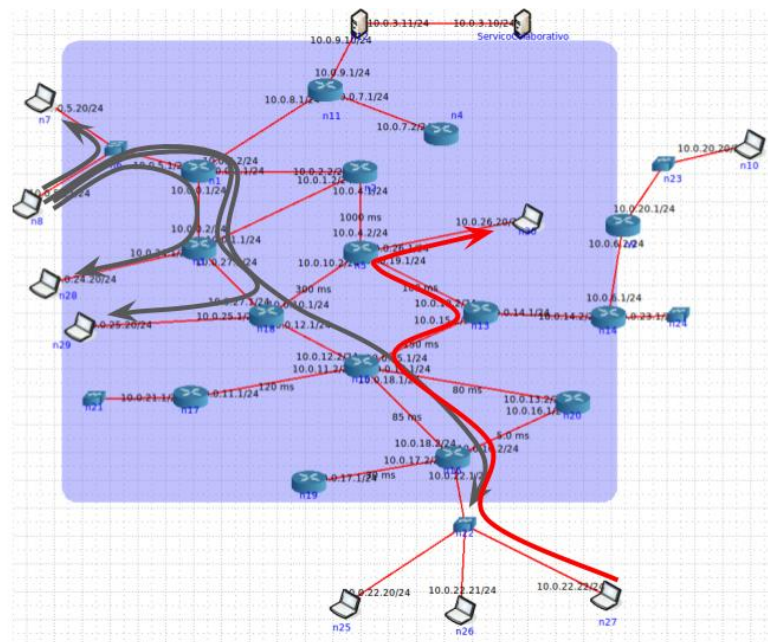


Figure 4.46.: Active Protection: Overlay Distribution Tree

4.5 LINK MINIMIZATION

In this test, the goal is not necessarily the enhancement of the overlay's performance but the reduction of the number of used links in the underlying network.

Now, as far as the testing conditions, Figure 4.2 represents, once more, the network topology being used for this testing stage. In order to validate the developed system, the following test sequence will be performed:

1. Activation of the scenario:
 - a) Activation of the Central Node;
 - b) Activation of the Collaborative Service;
 - c) Activation of the Sender (N8);
 - d) Activation of Receivers N7, N28, N29, N30, N25, N26, N27;
2. Collaborative Service will request the activation of link minimization procedures:
 - a) Central Node will perform the appropriate procedures;
 - b) Central Node will notify each peer of its new upstream and downstream list so changes can be applied where necessary.

4.5.1 *Activating the Scenario*

The activation of the scenario, as mentioned, is performed using the Minimum-Delay approach. As the modus-operandi for this system has been previously described, Figure 4.47 represents the multicast tree after the whole scenario has been activated.

Tree Stage: 7				
Sender	Receivers	Receiver/Transmitter	Receivers	Receiver
10.0.5.21 →	10.0.5.20 10.0.24.20 10.0.25.20 10.0.22.20 10.0.22.21 10.0.22.22	10.0.25.20 →	10.0.26.20	10.0.24.20
Receiver		Receiver		Receiver
10.0.22.20		10.0.26.20		10.0.22.22
Receiver		Receiver		
10.0.22.21		10.0.5.20		

Figure 4.47.: Scenario activation resulting Overlay Distribution Tree

4.5.2 *Link Minimization Procedures*

Analyzing Figure 4.47, with the minimum delay metric in use, and taking into consideration the topology represented, once more, in Figure 4.2, a few considerations can be made, when translating both these results into Figure 4.48:

1. Path (N8 - N7): 10.0.5.21 - 10.0.5.1 - 10.0.5.20 (2 links in use);
2. Path (N8 - N28): 10.0.5.21 - 10.0.5.1 - 10.0.0.2 - 10.0.24.20 (3 links in use);
3. Path (N8 - N29): 10.0.5.21 - 10.0.5.1 - 10.0.0.2 - 10.0.27.1 - 10.0.25.20 (4 links in use);
4. Path (N8 - N25/26/27): 10.0.5.21 - 10.0.5.1 - 10.0.0.2 - 10.0.27.1 - 10.0.12.2 - 10.0.18.2 - 10.0.22.20/21/22 (6 links in use per connection, meaning 18 links in use);
5. Path (N7 - N30): 10.0.5.20 - 10.0.5.1 - 10.0.2.2 - 10.0.4.2 - 10.0.26.20 (4 links in use).

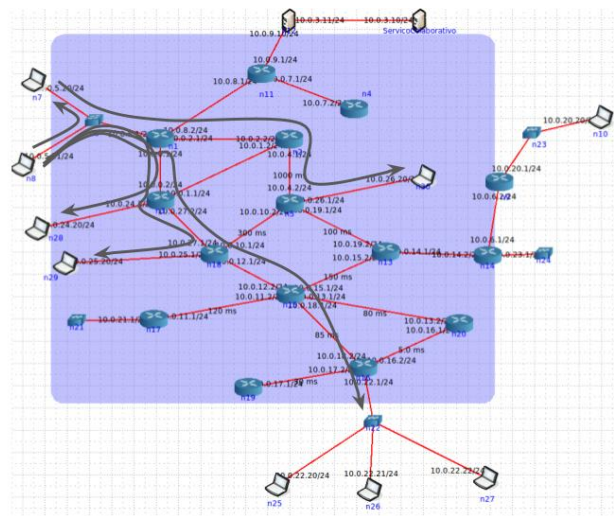


Figure 4.48.: Scenario activation result

The described paths, in the presented connections, bring the total count of used links in the multicast tree to 31 links (links being used more than once count separately). Now, when the ISP, via the collaborative service, requests the central node to perform the link minimization procedures, the minimum end-to-end delay for each peer logic is disregarded, and the goal becomes to have the number of links in use be as small as possible.

Now, when the Central Node receives the request from the collaborative service, it immediately starts performing the link minimization protocol. As seen in the protocol's description, this happens according to a set of steps:

1. Activating Link Parity;
2. Peer Indexation and Cost Association Graph;
3. Minimum-Spanning-Tree Generation (PRIM);
4. New Multicast Tree Formation;
5. Peer Notifications (if required).

Activating Parity Maps

Figure 3.13, in Section 3.3.2, shows the way parity maps information is represented. Essentially, it contemplates the best path from each peer in the session, to every other peer also in the session. It is a complete graph. Figure 4.49 shows a small representation of the process of creating a complete graph parity map, as it shows the central node determining the best path between any two peers. The presented process is performed for every single peer.

```

Path between: 10.0.5.20 and 10.0.22.22
(n1 - n3) -> (n3 - n18) -> (n18 - n15) -> (n15 - n16)
Path between: 10.0.5.20 and 10.0.22.21
(n1 - n3) -> (n3 - n18) -> (n18 - n15) -> (n15 - n16)
Path between: 10.0.5.20 and 10.0.5.21
Number of peers in the session: 8

```

Figure 4.49.: Parity Map Creation (summarized)

Peer Indexation and Cost Association

The current step has a rather simple job, which is to take the developed parity map, and translate it into a matrix, where rows and columns represent sources and destinations (Peer x Peer) and where each cell contains the cost of the best possible path between those two peers.

```

Printing Graph:
Line 0: 0 4 3 6 4 6 6 1
Line 1: 4 0 3 4 3 4 4 4
Line 2: 3 3 0 5 4 5 5 3
Line 3: 6 4 5 0 5 1 1 6
Line 4: 4 3 4 5 0 5 5 4
Line 5: 6 4 5 1 5 0 1 6
Line 6: 6 4 5 1 5 1 0 6
Line 7: 1 4 3 6 4 6 6 0

```

Figure 4.50.: Complete Graph Matrix

Figure 4.50 provides the visual representation of the matrix the central node created for the present case-study. It is clear to see that, paths already being used in the session in place, match the cost the central node calculates, which means, the central node accurately represented the session in the cost matrix.

Minimum-Spanning-Tree from PRIM

The deployment of PRIM, to determine the Minimum-Spanning-Tree, is according to the output of the previous step (which is the input matrix for the algorithm). When the algorithm is performed, the output is a vector, connecting indexed peers to their parent. The algorithm's output is represented in Figure 4.51.

So, with the Minimum-Spanning-Tree represented in Figure 4.51, the central node now simply has to derive the multicast tree from the peer index association with the presented children and corresponding parents.

```

Done with Prim Deployment, Time to implement multicast tree changes.
Printing Parent array
Child 0 with Parent -1
Child 1 with Parent 2
Child 2 with Parent 0
Child 3 with Parent 1
Child 4 with Parent 1
Child 5 with Parent 3
Child 6 with Parent 3
Child 7 with Parent 0

```

Figure 4.51.: PRIM output for MST

New Multicast Tree and Peer Notifications

Should the number of used links in the generated multicast tree be lesser than the number of links being used in the tree already in place, peers need to be notified to apply the proper changes. If this number is equal, no changes are made to the multicast tree, as there is no need to flood the network with peer notifications. In the present case-study, that necessity occurs.

Figure 4.52 shows the central node logging the fact that the resulting multicast tree is better than the one currently in place and, as such, peers need to be notified to apply the necessary changes so as to obtain the desired result. Also, each peer's acceptance of the changes is logged as well.

652	10.0.9.10	NoCentral	Changes have been made to the session with sender: 10.0.5.21. Notifying Peers.
655	10.0.9.10	NoCentral	Peer 10.0.25.20 acknowledged changes to the multicast tree.
658	10.0.9.10	NoCentral	Peer 10.0.24.20 acknowledged changes to the multicast tree.
662	10.0.9.10	10.0.9.10 - Porta Hello	Received Hello Message from Peer: 10.0.22.20
663	10.0.9.10	NoCentral	Peer 10.0.22.20 acknowledged changes to the multicast tree.
670	10.0.9.10	NoCentral	Peer 10.0.26.20 acknowledged changes to the multicast tree.
673	10.0.9.10	NoCentral	Peer 10.0.22.22 acknowledged changes to the multicast tree.
677	10.0.9.10	NoCentral	Peer 10.0.22.21 acknowledged changes to the multicast tree.
679	10.0.9.10	NoCentral	Peer 10.0.5.21 acknowledged changes to the multicast tree.
682	10.0.9.10	NoCentral	Peer 10.0.5.20 acknowledged changes to the multicast tree.

Figure 4.52.: Logging - Central Node logs peer Notifications and acknowledgements.

Now, Figure 4.53 shows the final multicast tree, after the procedure's execution and after peers have been notified. With this new multicast tree in mind, a similar reasoning to the one performed previously can be made:

1. Path (N8 - N7): 10.0.5.21 - 10.0.5.1 - 10.0.5.20 (2 links in use);
2. Path (N8 - N28): 10.0.5.21 - 10.0.5.1 - 10.0.0.2 - 10.0.24.20 (3 links in use);
3. Path (N28 - N29): 10.0.24.20 - 10.0.24.1 - 10.0.27.1 - 10.0.25.20 (3 links in use);
4. Path (N29 - N30): 10.0.25.20 - 10.0.25.1 - 10.0.10.2 - 10.0.26.20 (3 links in use);

- 5. Path (N29 - N25): 10.0.25.20 - 10.0.25.1 - 10.0.12.2 - 10.0.18.2 - 10.0.22.20 (4 links in use);
- 6. Path (N25 - N26): 10.0.22.20 - 10.0.22.1 - 10.0.22.21 (2 links in use);
- 7. Path (N25 - N27): 10.0.22.20 - 10.0.22.1 - 10.0.22.22 (2 links in use);

Which, brings the number of used links to a total of 19. Remember that the previous multicast tree used a total of 31 links.

Tree Stage: 8					
Sender	Receivers	Receiver/Transmitter	Receivers	Receiver/Transmitter	Receivers
10.0.5.21 →	10.0.24.20 10.0.5.20	10.0.25.20 →	10.0.22.20 10.0.26.20	10.0.24.20 →	10.0.25.20
Receiver/Transmitter	Receivers	Receiver		Receiver	
10.0.22.20 →	10.0.22.22 10.0.22.21	10.0.26.20		10.0.22.22	
Receiver		Receiver			
10.0.22.21		10.0.5.20			

Figure 4.53.: Final Multicast Tree.

Figure 4.54 provides a visual representation on the obtained results, where as few links as possible are used, with the desired goal having been obtained as initially 31 links were being used, and now only 19 are receiving traffic. Meaning, the underlying topology is much more protected, with a decrease of almost 40% in the number of used links.

4.6 ISP FORWARDER ACTIVATION

As mentioned in the description of the ISP Forwarder Activation methods, this approach requires action on the part of the ISP. As these actions come in the form of the activation of application level forwarders along the network, and given the testing platform present in all presented test cases, the way to simulate these forwarders is by appending such application entities to routers in the network. Thus, Figure 4.55 represents the new topology with most routers having "ovals" attached, where the ISP may execute and run applications.

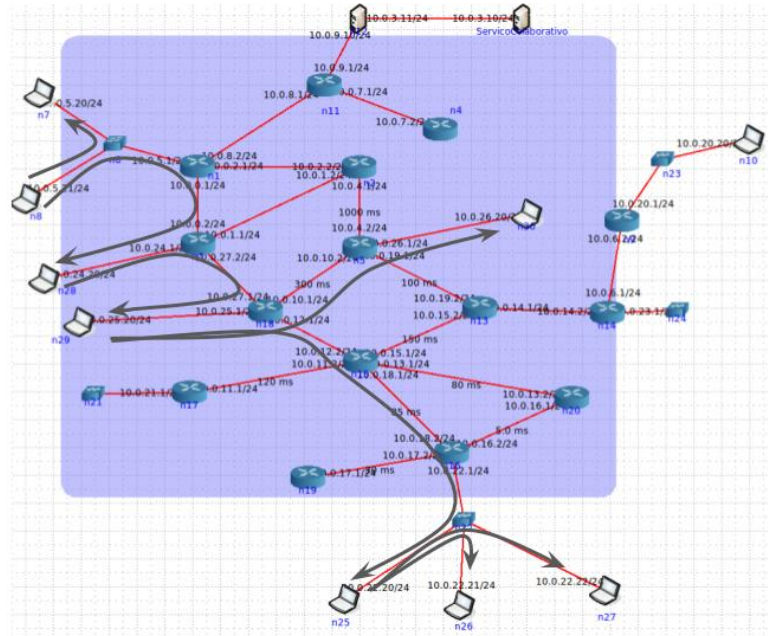


Figure 4.54.: Final multicast tree with demonstrated paths.

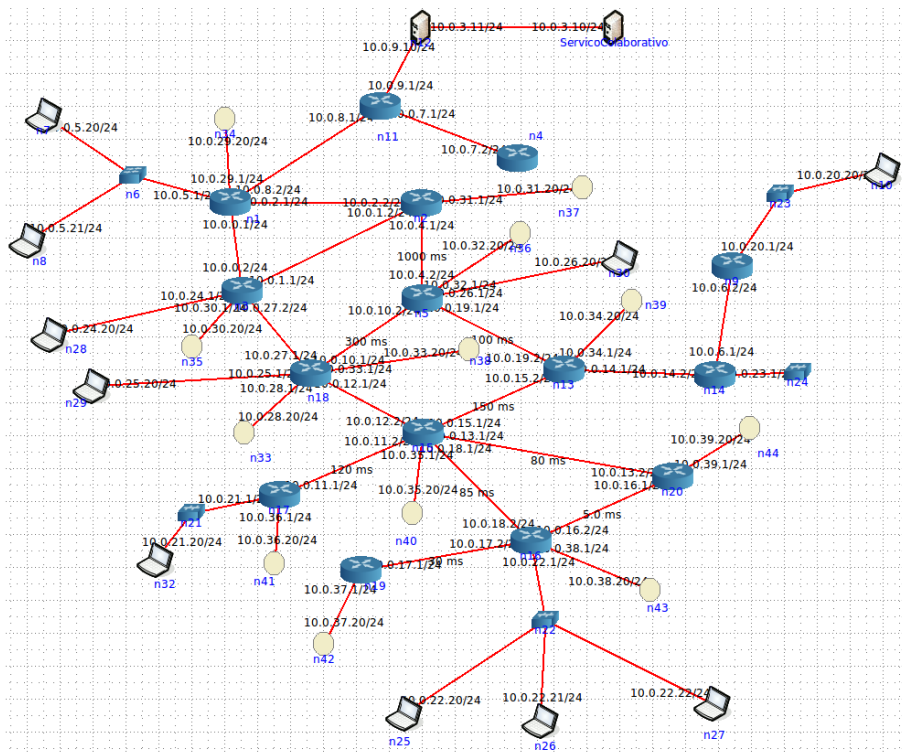


Figure 4.55.: Adapted Network Topology

The ISP is capable of activating any forwarder it deems necessary, at any time. And so, via the collaborative service, it is capable of communicating the addition of any new

forwarder to the central node, and having it test if the addition of this forwarder brings any possible improvements to the multicast sessions taking place.

Testing for the present usability context will happen according to the following steps:

1. Activation of the scenario (Minimum-Delay scenario, once more, will be used as the initial usability context):
 - a) Activation of the Central Node;
 - b) Activation of the Collaborative Service;
 - c) Activation of the Sender (N8);
 - d) Activation of Receivers N28, N30, N32, N25, N10;
2. The Collaborative Service will request the activation of link minimization procedures (as it is the starting point for forwarder improvement verification):
 - a) Central Node will perform the appropriate procedures;
 - b) Central Node will notify each peer of its new upstream peer and downstream list so changes can be applied where and if necessary;
3. The Collaborative Service will inform the Central node of the available forwarders, present in Figure 4.55, and will request it to elect one to use, if any. In the case where the inclusion of forwarders does not bring improvements to the network, no changes are made;
4. The Central Node:
 - a) will perform as many copies of the multicast session as available forwarders and include them in the multicast session;
 - b) per copy (and so per inclusion of each forwarder), will perform the Link Minimization procedures and determine the minimum-spanning tree including the forwarder as a peer;
 - c) per copy, will determine the number of links in use;
 - d) will elect the best case, whether it includes the addition of any forwarder to the multicast tree or not;
 - e) will notify each peer involved in the session of any changes to their upstream peer or downstream list, if any;
5. The Collaborative Service will activate the elected forwarder, if any.

4.6.1 *Activating the Scenario*

The scenario activation, again, is performed following the Minimum-Delay approach. With this usability context having been explained, demonstrated, and used previously, Figure 4.56 represents the obtained multicast tree with the scenario having been activated.

Tree Stage: 5				
Sender	Receivers	Receiver/Transmitter	Receivers	Receiver
10.0.5.21 →	10.0.24.20 10.0.21.20 10.0.22.20 10.0.26.20	10.0.22.20 →	10.0.20.20	10.0.24.20
Receiver		Receiver		Receiver
10.0.26.20		10.0.21.20		10.0.20.20

Figure 4.56.: Scenario activation result

4.6.2 *Forwarder Activation Step 1 - Link Minimization Procedures*

As described, the use of forwarders is to happen only in sessions where Link Minimization is the usability context taking place. To this point, the first step to be taken, upon activation of the scenario, is to have the ISP request that the central node perform the link minimization procedures to any active multicast session.

Taking a close look at Figure 4.56 and taking into consideration the adapted topology, Figure 4.55, allows, once more, for the conversion of these results into Figure 4.57.

Figure 4.57 allows for the determination of the paths being used, they have been shown in the previous sections.

Knowing the paths being used, it is easy to determine that the total number of links receiving traffic is 25. So, with the ISP requesting that the session be placed in the Link Minimization usability context, the previously described procedures are performed and a new multicast tree is generated, as shown in Figure 4.58.

Once more, from Figure 4.59 and the updated topology, the best path between each peer connection can be extrapolated, and so, the number of links receiving overlay traffic can be calculated. As these paths have also been demonstrated in the previous sections, it is clear the number of links in use is 21, 4 less than before.

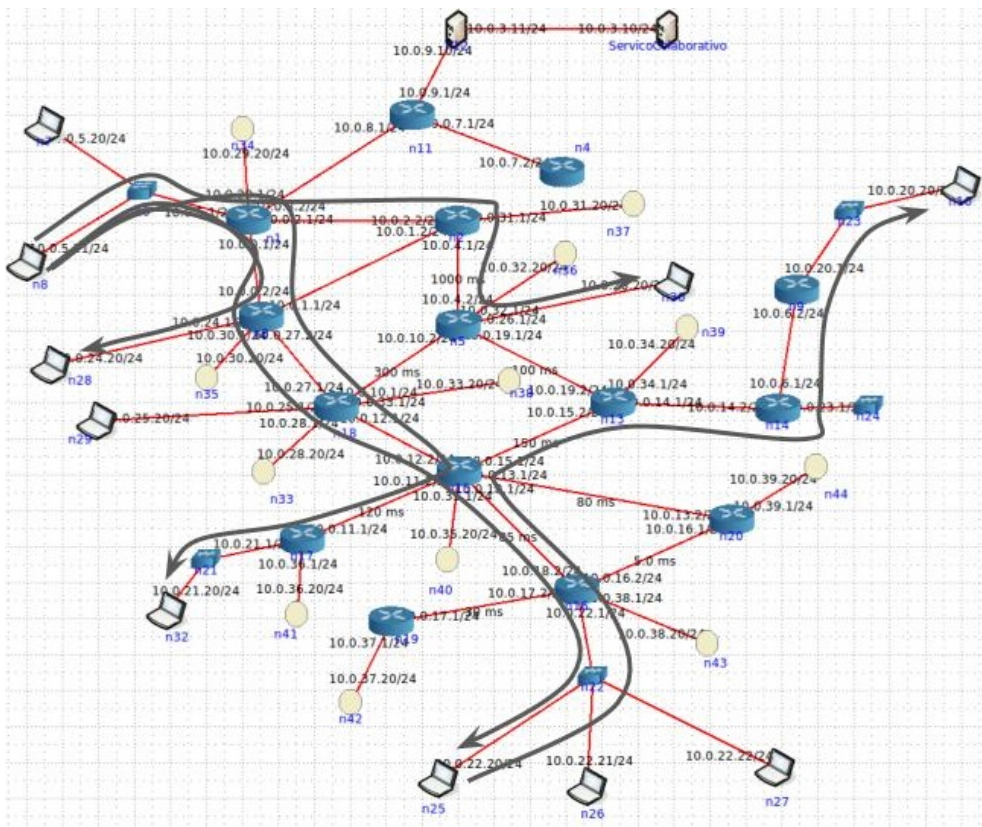


Figure 4.57.: Distribution Tree Representation

Tree Stage: 6					
Sender	Receivers	Receiver/Transmitter	Receivers	Receiver/Transmitter	Receivers
10.0.5.21	→ 10.0.24.20 10.0.26.20	10.0.24.20	→ 10.0.22.20	10.0.22.20	→ 10.0.21.20
Receiver/Transmitter	Receivers	Receiver		Receiver	
10.0.26.20	→ 10.0.20.20	10.0.21.20		10.0.20.20	

Figure 4.58.: Link Minimization result

4.6.3 Forwarder Activation Step 2 - Activating Forwarders

At this stage, the multicast session being tested, is employing a Link Minimization Scenario, as seen. So, the ISP is now able to request that it be tested for improvements, using the

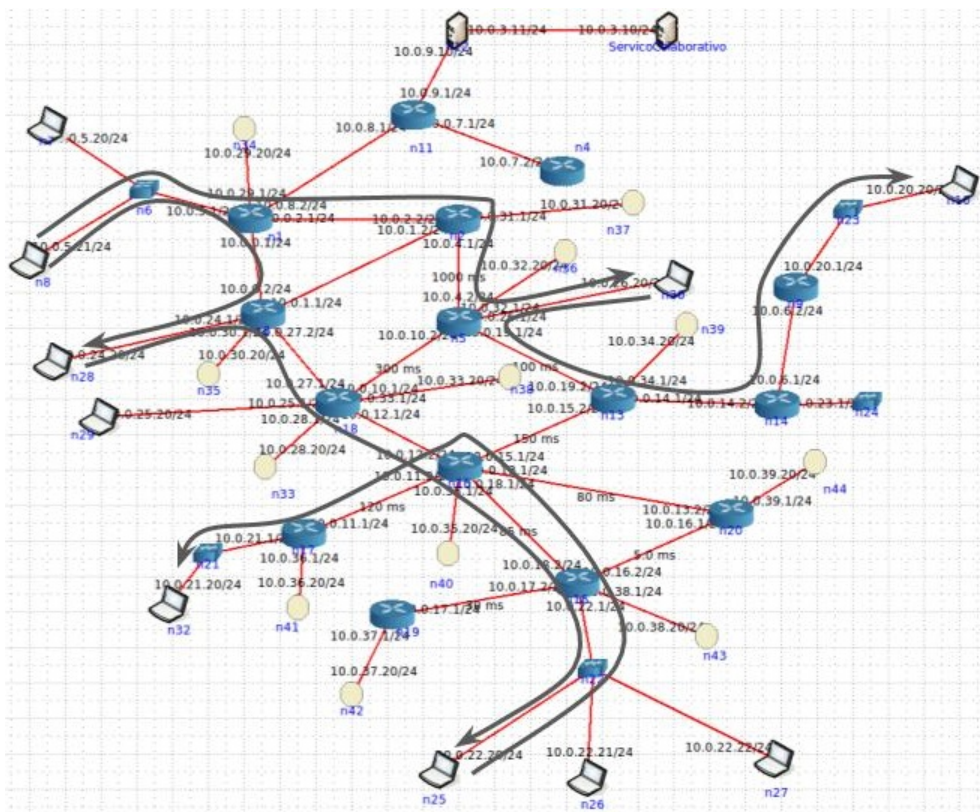


Figure 4.59.: Distribution Tree Representation 2

placed forwarders, seen in Figure 4.55. Now, as seen previously, all existent forwarders will be sent to the central node when the ISP requests the test, so, remembering the process:

1. The ISP will send the central node a request to have it test the session for improvements with the inclusion of forwarders;
2. The central node:
 - a) For each available forwarder, appends it to a copy of the session, and determines the number of links in use after performing the Link Minimization procedures with the added forwarder;
 - b) Assesses the best scenario and applies the necessary changes, if any.
3. The ISP activates the elected forwarder, if any.

With the process in mind, and having sent the request to the central node, Figure 4.60 represents the Central Node's report on performed simulations with the inclusion of each forwarder to the multicast tree. It becomes clear the current number of links being used is 21 (as demonstrated previously) and that only forwarder 10.0.37.20 would not result in

```

Current Tree: 21
10.0.37.20 - 23
10.0.34.20 - 18
10.0.33.20 - 18
10.0.39.20 - 21
10.0.29.20 - 20
10.0.35.20 - 17
10.0.38.20 - 20
10.0.36.20 - 20
10.0.28.20 - 18
10.0.31.20 - 20
10.0.32.20 - 19
10.0.30.20 - 19
DONE WITH REPORT
    
```

Figure 4.60.: Forwarder Tests Report

an improvement to the multicast tree. However, forwarder 10.0.35.20 provides the best outcome, as the number of used links when it is included in the multicast tree comes from 21 to 17, meaning a 4 link reduction.

So, the decision will be made to include the best forwarder into the multicast tree, 10.0.35.20, as seen in Figure 4.61.

Tree Stage: 7					
Sender	Receivers	Receiver/Transmitter	Receivers	Receiver/Transmitter	Receivers
10.0.5.21	→ 10.0.24.20	10.0.24.20	→ 10.0.35.20	10.0.35.20	→ 10.0.22.20 10.0.26.20 10.0.21.20 10.0.20.20
Receiver		Receiver		Receiver	
10.0.22.20		10.0.26.20		10.0.21.20	
Receiver					
10.0.20.20					

Figure 4.61.: Forwarder Activation result

It is, however, important to understand that the activation of forwarders along the network topology can result in much higher gains than presented, due to the fact that, in real scenarios, peers tend to be much more distributed along the network and not as concentrated as in the presented test case.

4.7 MULTIPLE AUTONOMOUS SYSTEMS

In section 3.4, an assessment is made on the way the developed prototype could operate under a wider scenario, i.e., one where peers are connected to the multicast sessions through

multiple autonomous systems. So, as to be able to test this characteristic, a new network topology is required, shown in Figure 4.62.

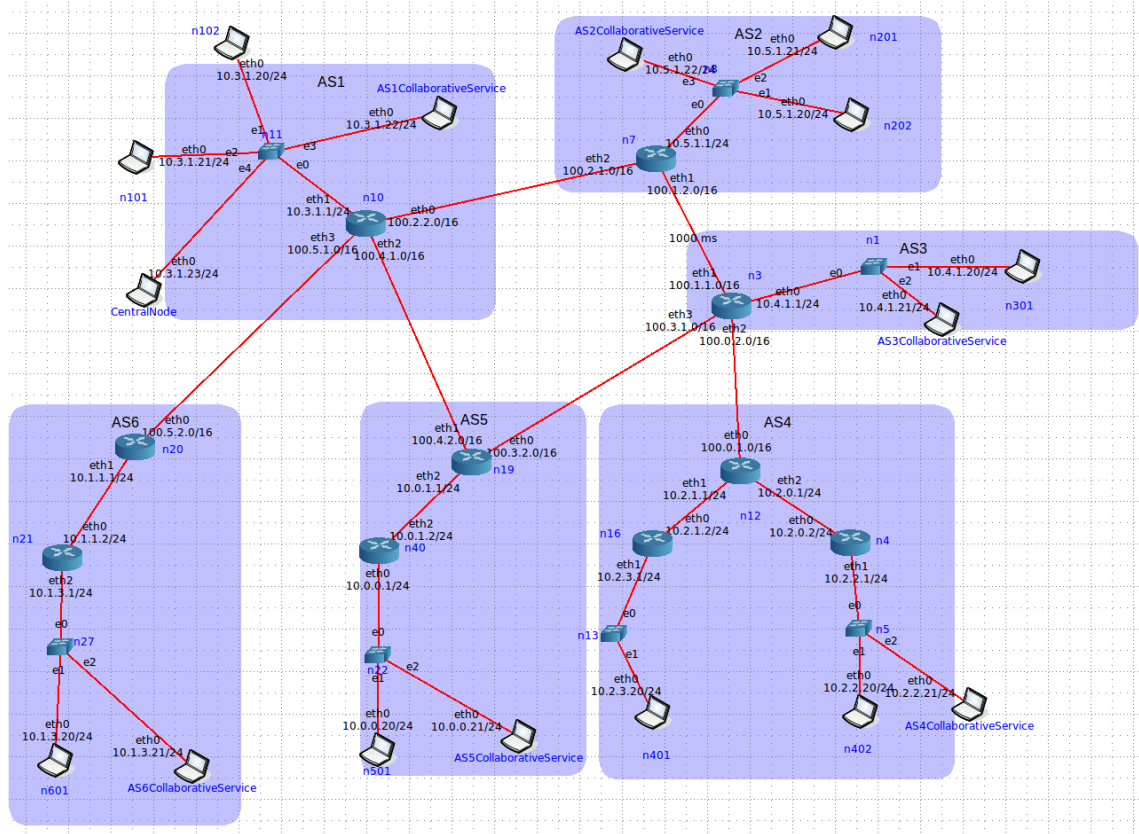


Figure 4.62.: Multiple Autonomous Systems Network Topology

The new topology allows testing under the described conditions, and contemplates, as mentioned, one Collaborative Service per Autonomous System. The necessity for one collaborative service per each AS has to do with the fact that only the ISP knows its own inner network topology as well as its external routing metrics, i.e, the path used to reach different Autonomous Systems, which is information the Central Node requires in order to be able to apply any desired changes to the overlay network.

So, knowing that each AS has its own collaborative service and that they all coordinate with the central node to achieve the desired goals, testing for the present usability context will happen according to the following steps:

1. Activation of the Central Node;
2. Activation of the Collaborative Service;
 - a) Each collaborative service will send its own inner network topology to the central node;

- b) Each collaborative service will send its external routes to the central node;
- 3. Activation of the Sender (N₁₀₁);
- 4. Activation of Receivers N₁₀₂, N₂₀₁, N₂₀₂, N₃₀₁, N₅₀₁, N₄₀₁, N₄₀₂, N₆₀₁;
- 5. One (or more in the future) of the Collaborative Services will request that the central node apply the inter-AS link minimization procedures;
- 6. The Central Node:
 - a) Will gather the information received from the different autonomous systems and emulate the topology they form;
 - b) Will apply the link minimization procedures to the external routing network (minimizing the usage of inter-AS links), meaning, the complete graph with costs between all peers will consider external paths only;
 - c) Per autonomous system, will elect a representative peer, which will receive external data and forward it to other peers in that specific autonomous system (this separation, allows for the future implementation of different usability contexts per autonomous system);
 - d) Will notify each peer involved in the session of any changes to their upstream peer or downstream list, if any.

4.7.1 *Activating the Scenario*

The scenario activation will be performed, once more, under the Minimum-Delay approach. Figure 4.63 shows a representation of the generated distribution tree.

It becomes clear that, even though multiple autonomous systems are considered, connections using inter-AS links are used as any other, with peers ignoring the fact that the usage of such links is much more costly. Now, in order to understand the way traffic is circulating, it is necessary to be given the external routing taking place:

- | | |
|---|--|
| <ul style="list-style-type: none"> 1. AS₁ <ul style="list-style-type: none"> • AS₂: n₇ (direct, 1 link); • AS₃: n₇ n₃ (2 links); • AS₄: n₇ n₃ n₁₂ (3 links); • AS₅: n₁₉ (direct, 1 link); • AS₆: n₂₀ (direct, 1 link); | <ul style="list-style-type: none"> 2. AS₂ <ul style="list-style-type: none"> • AS₁: n₁₀ (direct, 1 link); • AS₃: n₃ (direct, 1 link); • AS₄: n₃ n₁₂ (2 links); • AS₅: n₃ n₁₉ (2 links); • AS₆: n₁₀ n₂₀ (2 links); |
|---|--|

Tree Stage: 8				
Sender	Receivers	Receiver/Transmitter	Receivers	Receiver
10.3.1.21 →	10.3.1.20 10.5.1.21 10.5.1.20 10.4.1.20 10.0.0.20 10.1.3.20	10.0.0.20 →	10.2.3.20 10.2.2.20	10.3.1.20
Receiver		Receiver		Receiver
10.5.1.20		10.5.1.21		10.2.3.20
Receiver		Receiver		Receiver
10.2.2.20		10.1.3.20		10.4.1.20

Figure 4.63.: Scenario activation result

3. AS3

- AS1: n7 n10 (2 links);
- AS2: n7 (direct, 1 link);
- AS4: n12 (direct, 1 link);
- AS5: n19 (direct, 1 link);
- AS6: n7 n10 n20 (3 links);

5. AS5

- AS1: n10 (direct, 1 link);
- AS2: n3 n7 (2 links);
- AS3: n3 (direct, 1 link);
- AS4: n3 n12 (2 links);
- AS6: n10 n20 (2 links);

4. AS4

- AS1: n3 n7 n10 (3 links);
- AS2: n3 n7 (2 links);
- AS3: n3 (direct, 1 link);
- AS5: n3 n19 (2 links);
- AS6: n3 n7 n10 n20 (4 links);

6. AS6

- AS1: n10 (direct, 1 link);
- AS2: n10 n7 (2 links);
- AS3: n10 n7 n3 (3 links);
- AS4: n10 n7 n3 n12 (4 links);
- AS5: n10 n19 (2 links).

Although extensive, the presented information is the data the central node receives, per each autonomous system, in order to be able to create the necessary graph, required to perform its link minimization procedures. With this information in mind, and considering the presented generated distribution tree in Figure 4.63, it becomes clear that inter-AS links are consistently being used, which results in an unnecessary exploitation of resources.

4.7.2 Link Minimization Procedures and new Distribution Tree

At this point, the central node is considered to be topology-aware, i.e., it knows both the inner topology for each autonomous system as well as their external routing information. As mentioned, the central node is now able to apply the link minimization procedures considering the external paths only. Figure 4.64 shows the generated external graph matrix and Figure 4.65 provides a visual representation on the final multicast tree.

Generated Matrix

0	1	2	3	1	1
1	0	1	2	2	2
2	1	0	1	1	3
3	2	1	0	2	4
1	2	1	2	0	2
1	2	3	4	2	0

Figure 4.64.: External Graph Matrix

Tree Stage: 9					
Sender	Receivers	Receiver/Transmitter	Receivers	Receiver/Transmitter	Receivers
10.3.1.21 →	10.3.1.20 10.5.1.20 10.0.0.20 10.1.3.20	10.5.1.20 →	10.5.1.21 10.4.1.20	10.2.3.20 →	10.2.2.20
Receiver/Transmitter	Receivers	Receiver		Receiver	
10.4.1.20 →	10.2.3.20	10.3.1.20		10.5.1.21	
Receiver		Receiver		Receiver	
10.2.2.20		10.1.3.20		10.0.0.20	

Figure 4.65.: Final Distribution Tree

With the presented multicast tree, and considering also the external paths each autonomous system uses as well as the inner network topology, Figure 4.66 represents the used paths for data distribution, showing a reduction of the number of used inter-AS links as the link between AS₃ and AS₅ stops being required. Furthermore, a close analysis shows that only one copy of each datagram sent by the source's stream traverses each used inter-AS link, meaning, the process also reduced the number of datagram packets sent in inter-AS links.

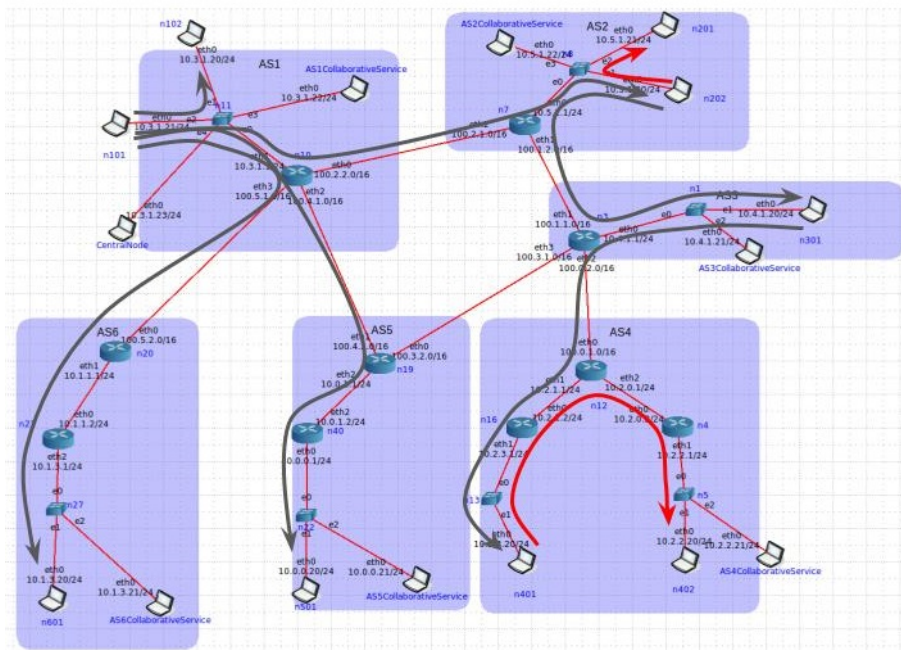


Figure 4.66.: Final distribution tree - Paths

CONCLUSIONS

This chapter will provide an overview on the developed work, both the envisioned architecture and associated mechanisms as well as the obtained results. Later, future work will be addressed as the developed prototype is extendable to more usability contexts and ways of operating.

5.1 DEVELOPED WORK

As presented in the first chapter of this document, this thesis' main goal consisted in the development of a Multicast Overlay network, adaptable to different usability contexts. Furthermore, one which is easily reconfigurable to assume different behaviours and metrics in the construction and management of the multicast distribution tree.

To the previous objectives, the first stage was the investigation of different technologies to provide the knowledge and sensibility to the requirements of such a system. The first studied technology was IP Multicast, a way of implementing group communication on the network level, the most efficient way even, but one with high operating costs and no pricing-model available, which has made the deployment of this technology rather slow and sparse. The second technology to be analyzed was the implementation of overlay networks on peer-to-peer systems. With the sparse deployment of multicast on the network level, solutions were presented in the application-level, where overlay networks are formed by peers, also called clients or end-hosts in many works, in which they act as both sources (providers of content) and clients (receivers of content). These peers participate together in the distribution of content throughout the overlay network they built on top of the underlying topology. Finally, Application-Layer Multicast was the third technology investigated, a technology that tries to bridge the two previously mentioned, as it implements group communication on an overlay network formed by peers which can be clients or even probes inserted by the ISPs themselves. This latter technology is the one closer to the developed architecture.

Having studied different approaches to the challenge in hand, the developed architecture consisted of three main entities: peer (sender or receiver), central node and the col-

laborative service (bridging the gap between the overlay's management and the ISP). Two methodologies were implemented which did not necessarily include the ISP's involvement (collaborative service), where the distribution tree was built to minimize either delay or loss, respectively, from source to sender. Furthermore, link protection, link minimization and forwarder activation were the other approaches and methodologies implemented sustained by using ISP collaborative approaches.

Finally, the testing stage consisted in the conception of a network topology in order to be able to simulate operating scenarios and test the architecture and methodologies implemented. This simulation was performed in CORE, a network emulator that accurately simulates real-life infrastructures and network systems. Also, this platform allows its users to run the developed code/projects, enabling proper testing. For better and simplified analysis on the developed work, all entities were made to log their decision-making processes and events, so a web application was also developed at this stage to gather data from all entities in the system, providing a simpler way to validate the obtained results.

With the presented development steps, it is important to note that the developed architecture and mechanisms operated as expected and the described objectives have been met. Peers can form the overlay network without any help from the collaborative service, i.e., the overlay operates independently whenever the ISP does not deem it necessary to intervene. Furthermore, the collaborative approaches have been shown to work properly, meaning, the ISP (via the collaborative service) can cooperate with the central node, requesting it to implement specific changes, such as link protection or link minimization, or they can even work together with the ISP activating forwarders along the topology in order to improve the system, forwarders whose activation the central node requests. The prototype has also been shown to operate correctly under scenarios considering multiple autonomous systems.

5.2 FUTURE WORK

Despite the fact that the objectives for this work have been achieved, the developed solution could be improved with further development.

While the system was made to work with multiple multicast sessions operating at the same time, sessions were made to be single-sourced so as to simplify and accelerate the development process. As future work, changing the multicast session's operation in order to support multiple senders will be an important issue. This could be made via the establishment of a distribution tree per source, or by having a single source tree, but one where other senders sent their data to that source, and it would distribute it throughout all the system, for instance.

Another aspect that could be improved with future work, has to do with the ISP Forwarder Activation methodology. The system contemplates only one forwarder at each time,

and could be extended to perform its improvement simulations with different combinations of various forwarders. This would ensure the absolute best possible result, and not search solely for the improvement of the system with the addition of one forwarder.

In the construction of the multicast distribution tree, methodologies have been implemented to work separately, i.e., the session either operates in one scenario or another, for instance, minimum-loss or minimum-delay. This decision, however, could be extended to hybrid approaches, where, for example, decisions could be made to minimize loss until a certain factor of loss percentage at which point, another route would be selected if possible. This more elaborated method would be a significant addition to the system as it would allow for the maximization of the system's performance. On this subject, extending the number of alternatives would also be important, such as bandwidth maximization, for instance.

BIBLIOGRAPHY

- [1] S. Deering, *Host Extensions for IP Multicasting*, RFC1112, <https://www.ietf.org/rfc/rfc1112.txt>, August 1989.
- [2] Jon Hardwick, *IP Multicast explained*, Metaswitch, <http://network-technologies.metaswitch.com/download/multicast.pdf>, 71, 2004.
- [3] Li Lao, Jun-Hong Cui, Mario Gerla, Dario Maggiorinni, *A Comparative Study of Multicast Protocols: Top, Bottom, or in the Middle?*, Proc. IEEE INFOCOM, vol. 4, pp. 2809 – 2814, 2005.
- [4] Suman Banerjee, Bobby Bhattacharjess, *A Comparative Study of Application Layer Multicast Protocols*, University of Maryland, 2002.
- [5] Miguel Castro, Peter Druschel, Y. Charlie Hu, Antony Rowstron, *Topology-Aware Routing in Structured Peer-to-Peer Overlay Networks*, Future Directions in Distributed Computing, part of, Lecture Notes in Computer Science, pp. 103-107, 2003.
- [6] Wojciech GalubaAffiliated, Sarunas Girdzijauskas, *Peer to Peer Overlay Networks: Structure, Routing and Maintenance*, Encyclopedia of Database Systems, pp 2056-2061, 2009.
- [7] Xinyan Zhang, Jiangchuan Liu, Bo Li, Tak-Shing Peter Yum, *CoolStreaming/DONet: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming*, Proc. IEEE INFOCOM, pp. 2102-2111, 2005.
- [8] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, Dan S. Wallach, *Secure routing for structured peer-to-peer overlay networks*, 5th Symposium on Operating Systems Design and Implementation, pp. 2909-314, 2002.
- [9] Apostolos Malatras, *State-of-the-art survey on P2P overlay networks in pervasive computing environments*, Journal of Network and Computer Applications, pp. 1-23, 2015.
- [10] Piotr Wydrych, Piotr Cholda, *ISP - Supported Traffic Reduction for Application-Level Multicast*, IEEE International Conference on Communications, pp. 1 – 6, 2011.

- [11] Fabio Picconi, Laurent Massoulié, *ISP Friend or Foe? Making P2P Live Streaming ISP-Aware*, IEEE International Conference on Distributed Computing Systems, pp. 413-422, 2009.
- [12] Li Lao, Jun-hong Cui, Mario Gerla, Shigang Chen, *A Scalable Overlay Multicast Architecture for Large-Scale Applications*, IEEE Transactions on Parallel and Distributed Systems, Vol. 4, pp. 449-459, 2007.
- [13] Kianoosh Mokhtarian, Hans-Arno Jacobsen, *Minimum-delay overlay multicast*, Proceedings IEEE INFOCOM, pp. 1771-1779, 2013.
- [14] Yang Cao, Xu Chen, Tao Jiang, Junshan Zhang, *SoCast: Social Ties Based Cooperative Video Multicast*, Proceedings INFOCOM, IEEE, pp. 415-423, April 2014.
- [15] B. Fenner, M. Handley, H. Holbrook, I. Kouvelas, *Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)*, RFC4601, <https://tools.ietf.org/html/rfc4601>, August 2006.
- [16] A. Adams, J. Nicholas, W. Siadak, *Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised)*, RFC3973, <https://tools.ietf.org/html/rfc3973>, January 2005.
- [17] M. Cotton, L. Vegoda, D. Meyer, *IANA Guidelines for IPv4 Multicast Address Assignments*, RFC5771, <https://tools.ietf.org/html/rfc5771>, March 2010.
- [18] H. Holbrook, B. Cain, *Source-Specific Multicast for IP*, RFC4607, <https://tools.ietf.org/html/rfc4607>, August 2006.
- [19] Eng Keong Lua, J. Crowcroft, M. Pias, R. Sharma, S. Lim, *A Survey and comparison of peer-to-peer overlay network schemes*, IEEE Communications Surveys and Tutorials, Vol. 7, pp. 72-93, 2005.
- [20] Vlachou, A., Doulkeridis, C., Norvag, K., Kotidis, Y., *Peer-to-Peer Query Processing over Multidimensional Data*, SpringerBriefs in Computer Science, Chapter 2, pp 5-12, 2012.
- [21] Rodrigo Rodrigues, Peter Druschel, *Peer-to-Peer Systems*, Communications of the ACM, Vol. 53 No. 10, pp. 72-82, October 2010.
- [22] Napster, available at <http://www.napster.com/>
- [23] Bittorrent, available at <http://www.bittorrent.com/>
- [24] Gnutella, described and available at <http://whatis.techtarget.com/definition/Gnutella>
- [25] Skype, available at <https://www.skype.com/>

- [26] Stanford University, Folding Coin, available at <http://foldingcoin.net/>
- [27] UC Berkeley, SETI@home, available at <https://setiathome.berkeley.edu/>
- [28] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, Hari Balakrishnan, *Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications*, SIGCOMM, Conference on Applications, technologies, architectures, and protocols for computer communications, pp. 149-160, 2001.
- [29] Nabhendra Bisnik, Alhussein A. Abouzeid, *Optimizing random walk search algorithms in P2P networks*, Computer Networks, Volume 51, Issue 6, pp. 1499-1514, 2007.
- [30] Pedro Sousa, *Towards Effective Control of P2P Traffic Aggregates in Network Infrastructures*, Journal of Communications Software and Systems, 11(1), 37-47, 2015.
- [31] Pedro Sousa, *A Framework for Highly Reconfigurable P2P Trackers*, Journal of Communications Software and Systems, 9(4), 236-246, 2013.
- [32] Pedro Sousa, *Traffic Engineering Approaches in P2P Environments*, 5th International Conference on Advanced Infocomm Technology (ICAIT 2012), Paris, France, Springer, LNCS 7593, pp. 61-74, 2013.
- [33] John Jannotti, David K. Gifford, Kirk L. Johnson, M. Frans Kaashoek, James W. O'Toole JR., *Overcast: Reliable Multicasting with an Overlay Network*, Proc. Of the 4th conference on Symposium on Operating System Design and Implementation, vol. 4, pp. 197-212, 2000.
- [34] Suman Banerjee, Christopher Kommareddy, Koushik Kar, Bobby Bhattacharjee, Samir Khuller, *Construction of an Efficient Overlay Multicast Infrastructure for Real-time Applications*, IEEE Societies INFOCOM, vol. 2, pp. 1521 – 1531, 2003.
- [35] Sherlia Y. Shi, Jonathan S. Turner, Marcel Waldvogel, *Dimensioning server access bandwidth and multicast routing in overlay networks*, NOSSDAV: Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video, pp. 83-91, 2001.
- [36] Yatin Chawathe, Steven McCanne, Eric Brewer, *An Architecture for Internet Content Distribution as an Infrastructure Service*, Ph.D. Thesis. University of California, Berkeley 2000.
- [37] L. Lao, J.-H. Cui, and M. Gerla, *Multicast Service Overlay Design*, Proc. of International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'05), Philadelphia, PA, USA, July 2005.

- [38] Vinay Aggarwal, Anja Feldmann, Christian Scheideler, *Can ISPs and P2P Users Cooperate for Improved Performance?*, ACM SIGCOMM Computer Communication Review, Volume 37 Issue 3, pp. 29-40, July 2007.
- [39] S.Y. Shi, J.S. Turner, *Routing in Overlay Multicast Networks*, Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies, pp. 1200-1208, 2002.
- [40] J. Zhao, F. Yang, Q. Zhang, Z. Zhang, F. Zhang, *LION: Layered Overlay Multicast With Network Coding*, IEEE Transactions on Multimedia, Vol. 8, pp. 1021-1032, 2006.
- [41] Zhou Su, Masato Oguro, Yohei Okada, Jiro Katto, Sakae Okubo, *Overlay tree construction to distribute layered streaming by application layer multicast*, IEEE Transactions on Consumer Electronics, Vol. 56, pp. 1957-1962, 2010.
- [42] Reza Besharati, Mozafar Bag-Mohammadi, Mashallah Abbassi Dezfouli, *A Topology-Aware Application Layer Multicast Protocol*, Consumer Communications and Networking Conference (CCNC), IEEE, pp. 1-5, 2010.
- [43] Suman Banerjee, Bobby Bhattacharjee, Christopher Kommareddy, *Scalable application layer multicast*, Proceedings of conference on Applications, technologies, architectures, and protocols for computer communications, pp. 205-217, 2002.
- [44] Kai-Wei Ke, Chia-Hui Huang, *Performance evaluation of multisource Application Layer Multicast (ALM): Theoretical and simulative aspects*, Computer Networks, Volume 57, Issue 6, pp. 1408-1424, 2013.
- [45] Mojtaba Hosseini, Dewan Tanvir Ahmed, Shervin Shirmohammadi, Nicolas D. Georganas, *A Survey of Application-Layer Multicast Protocols*, IEEE Communications Surveys & Tutorials, Vol. 9, Issue 3, pp. 58-74, 2007.
- [46] Mirja Kühlewind, Jan Seedorf, Vijay Gurbani, *ALTO Status Page*, IETF Group, <https://datatracker.ietf.org/wg/alto/charter/>.
- [47] Graph Library, <http://algs4.cs.princeton.edu/41graph/SymbolGraph.java.html>, as seen in 2017.
- [48] PRIM algorithm, <http://www.geeksforgeeks.org/greedy-algorithms-set-5-prims-minimum-spanning-tree-mst-2/>, as seen in 2017.
- [49] Graphs data structures for library, <http://algs4.cs.princeton.edu/>, as seen in 2017
- [50] CORE network emulator, <https://www.nrl.navy.mil/itd/ncs/products/core>, version 4.8.



SECONDARY TREE REPRESENTATION

A.1 MINIMUM-DELAY APPROACH

```
INFO: -----Sender: 10.0.5.21 -----  
10.0.25.20 - 10.0.26.20 -  
10.0.24.20 -  
10.0.22.20 -  
10.0.26.20 -  
10.0.22.22 -  
10.0.22.21 -  
10.0.5.21 - 10.0.5.20 - 10.0.22.20 - 10.0.22.21 - 10.0.22.22 - 10.0.24.20 - 10.0.25.20 -  
10.0.5.20 -  
-----
```

Figure A.1.: Receivers Activation Representation 2

```
INFO: -----Sender: 10.0.5.21 -----  
10.0.25.20 - 10.0.26.20 -  
10.0.26.20 -  
10.0.22.22 -  
10.0.22.21 -  
10.0.5.21 - 10.0.5.20 - 10.0.22.21 - 10.0.22.22 - 10.0.25.20 -  
10.0.5.20 -  
-----
```

Figure A.2.: Deactivating Node N25 - Tree Representation 2

A.2 PASSIVE LINK PROTECTION

```
INFO: -----Sender: 10.0.5.21 -----  
10.0.22.20 - 10.0.22.21 - 10.0.22.22 -  
10.0.22.22 -  
10.0.22.21 -  
10.0.5.21 - 10.0.5.20 - 10.0.22.20 -  
10.0.5.20 -  
-----
```

Figure A.3.: Multicast Tree Stage 4 -Representation 2

