

# HYBRID MULTICASTING USING AUTOMATIC MULTICAST TUNNELS (AMT)

DHAIFALLAH ALWADANI



Doctor of Philosophy

Institute of Computing Science and Mathematics

University of Stirling

March 2017



## DECLARATION

---

I, Dhaifallah Alwadani, hereby declare that the work in this thesis is original and has been composed by myself, except where reference is made to other works, and has not been submitted for examination for any other degree at this university or any other learning institutions.

*Stirling, March 2017*

---

Dhaifallah Alwadani



## ABSTRACT

---

Native Multicast plays an important role in distributing and managing delivery of some of the most popular Internet applications, such as IPTV and media delivery. However, due to patchy support and the existence of multiple approaches for Native Multicast, the support for Native Multicast is fragmented into isolated areas termed Multicast Islands. This renders Native Multicast unfit to be used as an Internet wide application. Instead, Application Layer Multicast, which does not have such network requirements but is more expensive in terms of bandwidth and overhead, can be used to connect the native multicast islands. This thesis proposes Opportunistic Native Multicast (ONM) which employs Application Layer Multicast (ALM), on top of a DHT-based P2P overlay network, and Automatic Multicast Tunnelling (AMT) to connect these islands. ALM will be used for discovery and initiating the AMT tunnels. The tunnels will encapsulate the traffic going between islands' Primary Nodes (PNs). AMT was used for its added benefits such as security and being better at traffic shaping and Quality Of Service (QoS). While different approaches for connecting multicast islands exists, the system proposed in the thesis was designed with the following characteristics in mind: scalability, availability, interoperability, self-adaptation and efficiency. Importantly, by utilising AMT tunnels, this approach has unique properties that improve network security and management.



## ACKNOWLEDGMENTS

---

For me, undertaking this PhD has been a truly life-changing experience and it would not have been possible to do without the help and support that I received.

First of all, I am extremely thankful to Almighty Allah for his blessings and providing me with the ability to carry out this research, without which none of my work would have been possible.

My sincerest gratitude and deepest appreciation goes to my supervisor Dr. Mario Kolberg for his guidance throughout my PhD study. Without his guidance and constant feedback this PhD would not have been achievable.

I would also like to express my thanks to my parents, Bakhit and Haya, for their support, prayers, advice and encouragement throughout my research and without which, I would not have had the courage to embark on this journey in the first place. Their prayers and belief have been an amazing source of comfort.

Also, many thanks to my brothers and sisters: Norah, Layla, Eman, Sarah, Mubarak, Amer, Abdulrahman and my little sister Lamia.

I would also like to thank my beloved wife, Hajar, for her endless patience, continuous encouragement, and support and for being by my side throughout this PhD, living every single minute of it.

Finally, my love to my daughter, Haya, who was a constant source of joy pushing me forward.





## LIST OF PUBLICATIONS

---

During the period of this research, the following papers have been published:

- D. Alwadani, M. Kolberg, and J. Buford, "A Simulation Model for Hybrid Multicast," 2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies, pp. 112-116, 2014. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6982901>
- D. Alwadani, M. Kolberg, and J. Buford, "An evaluation of opportunistic native multicast," Computer Aided Modelling and Design of Communication Links and Networks (CAMAD), 2015 IEEE 20th International Workshop on, pp. 170-174, 2015.
- D. Alwadani, M. Kolberg, and J. Buford, "Opportunistic native multicast under churn," in SAI Computing Conference (SAI), 2016. IEEE, 2016, pp. 644-648.
- D. Alwadani and M. Kolberg. (2017). Opportunistic Native Multicast. Submitted to International Journal of Parallel, Emergent and Distributed Systems



# CONTENTS

---

1	INTRODUCTION	1
1.1	Background . . . . .	1
1.2	Research Problem . . . . .	3
1.3	Thesis Statement . . . . .	4
1.4	Aims and Objectives . . . . .	4
1.5	Contributions . . . . .	5
1.6	Thesis Scope and Structure . . . . .	7
2	BACKGROUND AND RELATED WORK	9
2.1	Multicast . . . . .	9
2.1.1	Taxonomy . . . . .	11
2.1.2	Native Multicast . . . . .	15
2.1.2.1	Host Group Multicast . . . . .	15
2.1.2.2	Multi-Destination Routing . . . . .	17
2.1.3	Application Layer Multicast . . . . .	18
2.1.3.1	CAN-Multicast . . . . .	19
2.1.3.2	SCRIBE . . . . .	20
2.1.4	Hybrid Multicast . . . . .	21
2.1.4.1	Automatic Multicast Tunnelling AMT . . . . .	22
2.1.4.2	Island Multicast . . . . .	23
2.1.4.3	Universal Multicast . . . . .	23
2.1.4.4	Multicast Delivery Based on Unicast and Subnet Multicast . . . . .	24
2.1.4.5	Hybrid Multicast Issues . . . . .	24
2.2	Peer To Peer . . . . .	26
2.2.1	Taxonomy . . . . .	27
2.2.2	Structured Peer-to-Peer Overlay . . . . .	28
2.2.3	Types of Structured Overlays . . . . .	29

2.2.3.1	Multi-Hop . . . . .	29
2.2.3.2	One Hop . . . . .	35
2.2.3.3	Variable Hop . . . . .	37
2.2.4	Unstructured . . . . .	38
2.2.5	Differences between Structured and Unstructured Overlays .	39
2.3	Automatic Multicast tunnelling (AMT) . . . . .	40
2.3.1	AMT Operation . . . . .	41
2.3.2	Advantages of AMT . . . . .	42
2.4	Summary . . . . .	44
3	OPPORTUNISTIC NATIVE MULTICAST	45
3.1	Introduction . . . . .	45
3.1.1	Joining ONM . . . . .	46
3.1.2	ONM Alternatives . . . . .	48
3.1.3	ONM Advantages . . . . .	49
3.2	ONM Operation . . . . .	50
3.3	Primary Election . . . . .	53
3.4	Secondary Selection . . . . .	56
3.5	Failure Recovery . . . . .	58
3.6	Summary . . . . .	62
4	PERFORMANCE EVALUATION OF ONM	63
4.1	Introduction . . . . .	63
4.2	Experimental Methodology . . . . .	63
4.2.1	Research Questions . . . . .	63
4.2.2	Benchmark Selection . . . . .	64
4.2.3	OMNet++ and INET Framework . . . . .	65
4.2.4	OverSim . . . . .	66
4.2.5	Simulation Setup . . . . .	67
4.2.6	Simulation Model . . . . .	68
4.3	AMT . . . . .	70
4.3.1	Changes to The Simulation Environment . . . . .	70
4.3.1.1	AMT Gateway . . . . .	70

4.3.1.2	AMT Relay . . . . .	71
4.3.1.3	Changes in OverSim . . . . .	73
4.3.1.4	Network Messages . . . . .	74
4.3.2	The Network Model . . . . .	75
4.4	Basic Implementation . . . . .	75
4.4.1	Overview . . . . .	75
4.4.2	Simulation Scenarios . . . . .	76
4.4.2.1	Small proof-of-concept network . . . . .	76
4.4.2.2	Applications . . . . .	76
4.5	ONM Vs ALM . . . . .	78
4.6	Number of Secondaries . . . . .	82
4.6.1	Introduction . . . . .	82
4.7	Approaches To Select Primary and Secondary Nodes . . . . .	86
4.7.1	Heterogeneity . . . . .	86
4.7.2	Distinguishing Low Churn Nodes . . . . .	87
4.7.3	Manual Priority . . . . .	89
4.7.4	Passive Priority . . . . .	92
4.7.5	Age-Based Priority . . . . .	92
4.7.6	Comparing Types of Priority . . . . .	97
4.8	Dynamic Heartbeat Intervals . . . . .	99
4.8.1	Dynamic Control Interval . . . . .	99
4.8.2	Probation Period . . . . .	100
4.8.3	Graduate Trust . . . . .	101
4.9	Detection of Node Failure . . . . .	105
4.9.1	Introduction . . . . .	105
4.9.2	Heartbeat Timeout Timer . . . . .	105
4.9.3	Selecting the value of N . . . . .	106
4.10	Comparing ONM with IM . . . . .	108
4.10.1	Performance Analysis . . . . .	108
4.11	Summary and Conclusion . . . . .	109
5	CONCLUSION AND FUTURE WORK . . . . .	115

5.1	Introduction . . . . .	115
5.2	Contributions . . . . .	116
5.2.1	Review of current literature . . . . .	116
5.2.2	Connecting Multicast Islands using AMT and ALM . . . . .	116
5.2.3	Detect Nodes Failure . . . . .	117
5.2.4	Availability and resilience . . . . .	117
5.2.5	Distinguishing between low and high churn nodes in a heterogeneous network . . . . .	118
5.2.6	Dynamically Optimising control traffic . . . . .	118
5.2.7	Performance of ONM . . . . .	118
5.2.8	The simulator . . . . .	119
5.3	Limitations and Future Work . . . . .	119
5.4	Summary . . . . .	121

## LIST OF FIGURES

---

Figure 1.1	Example of Multicast Islands . . . . .	2
Figure 2.1	a) Unicast, b) IP Multicast, c) Application Layer Multicast.[1]	10
Figure 2.2	Multicast Taxonomy . . . . .	15
Figure 2.3	HGM Multicasting . . . . .	16
Figure 2.4	IGMP Message . . . . .	17
Figure 2.5	MDR Multicasting . . . . .	18
Figure 2.6	A Sample Hybrid Multicast Network. [2] . . . . .	21
Figure 2.7	Example of routing a query from node 0132 to node 3121 in Pastry . . . . .	31
Figure 2.8	Example of the finger table in Chord for node a . . . . .	33
Figure 2.9	Kademlia Routing Buckets for node 110 . . . . .	34
Figure 2.10	The CAN overlay divided into zone in a 2 dimensional co- ordinate space . . . . .	35
Figure 2.11	An Example of an event propagating using Event Detection and Report Algorithm (EDRA) in D1HT . . . . .	37
Figure 2.12	The Flow of AMT messages . . . . .	43
Figure 2.13	Multicast Islands Connected Using AMT . . . . .	43
Figure 3.1	ONM System Topology . . . . .	46
Figure 3.2	An Island Joining ONM . . . . .	47
Figure 3.3	The stack of protocol used in ONM . . . . .	52
Figure 3.4	Primary Node Election Message Sequence . . . . .	54
Figure 3.5	Message Sequence for Selecting a Secondary Node (SN) . . .	57
Figure 3.6	The Message Sequence for Recovering after The Churn of The PN . . . . .	59
Figure 3.7	The Message Sequence for Recovering after The Churn of The SN . . . . .	60

Figure 3.8	ONM Operation FSM . . . . .	61
Figure 4.1	OverSim Structure . . . . .	66
Figure 4.2	A view on the design of Standard Host component in INET .	69
Figure 4.3	A view on the design of Standard Host component in INET .	69
Figure 4.4	The stack of components in AMT Gateway . . . . .	71
Figure 4.5	The stack of components in AMT Relay . . . . .	72
Figure 4.6	A dialog for the number of multicast islands . . . . .	75
Figure 4.7	A dialog for the number of client that will be randomly scattered across different islands . . . . .	76
Figure 4.8	An example network with two multicast-enabled islands . .	77
Figure 4.9	The AMT Application . . . . .	77
Figure 4.10	Comparing the Stretch of different multicast approaches . . .	79
Figure 4.11	Comparing the Stress of different multicast approaches . . .	80
Figure 4.12	Comparing the traffic generated in each island for different multicast approach . . . . .	80
Figure 4.13	Comparing the traffic crossing the backbone for different multicast approach . . . . .	81
Figure 4.14	Comparing the delay for different multicast approaches . . .	82
Figure 4.15	The Effect of lifetime and heartbeat intervals on Success Rate	84
Figure 4.16	The Effect of lifetime and heartbeat intervals on Overhead . .	85
Figure 4.17	The Effect of the Number of Secondary Nodes on Success Rate (With Heartbeat = 10) . . . . .	85
Figure 4.18	The Effect of the Number of Secondary Nodes on Overhead (With Heartbeat = 10) . . . . .	86
Figure 4.19	The Effect of manual selecting of Primary Nodes Success rate	90
Figure 4.20	The Effect of manual selecting of Primary Nodes on Overhead	91
Figure 4.21	The Effect of the Length of the simulation time on Success rate	93
Figure 4.22	The Effect of the Length of the simulation time on Overhead	94
Figure 4.23	The Effect of factoring age on Success rate . . . . .	95
Figure 4.24	The Effect of Factoring Age on Overhead . . . . .	96



Figure 4.25	The Effect of different methods of selecting the Primary Nodes with different heartbeat frequencies and number of Secondaries . . . . .	98
Figure 4.26	The Effect of Probation-time Dynamic Interval . . . . .	102
Figure 4.27	The Effect of Dynamic Interval in Graduate Trust . . . . .	103
Figure 4.28	The effect of heartbeat timeout value . . . . .	107
Figure 4.29	Comparing Island Multicast (IM) with Opportunistic Native Multicast (ONM) . . . . .	108



## LIST OF TABLES

---

Table 2.1	Conceptual comparison of IP multicast and ALM . . . . .	9
Table 2.2	Comparing Different Types of Structured Peer-To-Peer Networks . . . . .	29
Table 2.3	Example of Pastry Routing Table for a node with ID 12030213 and $b = 2$ . . . . .	30
Table 2.4	Comparing Different Types of Unstructured Peer-To-Peer Networks . . . . .	38
Table 3.1	The ONM Timers . . . . .	59
Table 4.1	List of default parameters for the simulations . . . . .	67
Table 4.2	Example content of an AMT-Gateway Peer Table . . . . .	71
Table 4.3	Example content of an AMT-Relay Peer Table . . . . .	74
Table 4.4	Parameters for $T_{Transition}$ . . . . .	101
Table 4.5	Calculation of $C$ . . . . .	104
Table 4.6	The chance of unneeded takeovers for different values of $N$ and Packet Drops . . . . .	107
Table 4.7	Success Rate of different configurations of Lifetime, Number of Secondaries and Primary Selection Method . . . . .	111
Table 4.8	Average overhead of different configurations of Lifetime, Number of Secondaries and Primary Selection Method . . . . .	112
Table 4.9	The Average delay of the multicasted messages . . . . .	113
Table 4.10	Average Number of messages in the island . . . . .	113
Table 4.11	Average stress on the backbone . . . . .	114
Table 4.12	Average stretch of the multicasted message . . . . .	114



## LIST OF ACRONYMS

---

<b>ALM</b>	Application Layer Multicast
<b>AMT</b>	Automatic Multicast Tunnelling
<b>AMT-GW</b>	AMT Gateway
<b>CAN</b>	Content Addressable Network
<b>CIM</b>	Centralised Island Multicast
<b>DHT</b>	Distributed Hash Table
<b>DIM</b>	Distributed Island Multicast
<b>EDRA</b>	Event Detection and Report Algorithm
<b>GRE</b>	Generic Routing Encapsulation
<b>HGM</b>	Host Group Multicasting
<b>HM</b>	Hybrid Multicast
<b>IGMP</b>	Internet Group Management Protocol
<b>IoT</b>	Internet of Things
<b>IPTV</b>	Internet Protocol TeleVision
<b>IM</b>	Island Multicast
<b>MAC</b>	Message Authentication Code
<b>MDR</b>	Multi-Destination Routing
<b>NED</b>	Network Description

**NM** Native Multicast

**ONM** Opportunistic Native Multicast

**OSI** Open Systems Interconnection

**P2P** Peer To Peer

**PN** Primary Node

**PRR** Plaxton, Rajaraman and Richa

**QoS** Quality Of Service

**SEPastry** Security Enhanced Pastry

**SHA1** Secure Hash Algorithm 1

**SN** Secondary Node

**TTL** Time To Live

**UDP** User Datagram Protocol

**UM** Universal Multicast

**VoIP** Voice Over Internet Protocol

**XOR** Exclusive OR

## INTRODUCTION

---

### 1.1 BACKGROUND

According to Cisco's Visual Networking Index [3], 91% of the Internet traffic is expected to be video. Multicast can play an important role in distributing and managing it. Internet applications, such as Internet Protocol TeleVision (IPTV) and multimedia conference calls, rely on distributing content in a one-to-many or many-to-many approach. This way of content delivery is called multicasting. Multicasting is a very powerful and efficient way to deliver content in the Internet. It was designed to save bandwidth and manage the routing and delivering of content to multiple destinations efficiently.

Multicast can be implemented at different layers of the network stack: mainly on the Network Layer and the Application Layer. If it is implemented at the network layer, referred to as Native Multicast (NM), routers forward and replicate the multicast messages. In this case, routers form a spanning tree for each multicast group. Alternatively, with Application Layer Multicast (ALM), hosts, not routers, are responsible for forming the spanning tree and to replicate and forward the multicast messages.

In NM, two main approaches can be used: Host Group Multicasting (HGM) and Multi-Destination Routing (MDR). In HGM, the routers can keep track of the multicast group. They are responsible for registering interested hosts. Also, they keep track of the group state. The sender needs to specify which group the message is for and the network takes care of the rest. Alternatively, in MDR, the source of the message needs to specify the destination hosts in the header of the message. After that, the routers duplicate the data, split the header and forward the message as needed.

Crucially, ALM does not require multicast support from routers. However, it is less efficient than Native Multicast as it sends multiple copies of the same message across the same link [4, 5]. On the other hand, to be able to use native multicasting, the routers need to be multicast-capable. Currently, this is not the case universally across the entire Internet [6]. This has resulted in multicast-capable parts of the Internet being scattered across the network as islands as shown in Figure 1.1. Furthermore, different native multicast technologies exist, such as HGM and MDR [7]. While the spread of the use of multicast-capable routers is increasing, the problem of multicast islands persists. Without approaches which bridge the islands, content providers rely on unicast to distribute content until global adoption of a (single) native multicasting approach is achieved. One

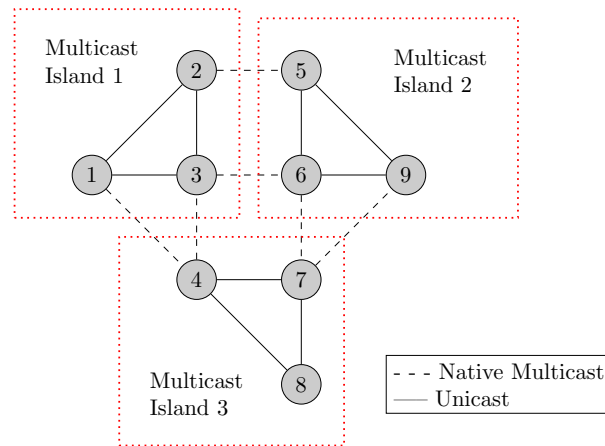


Figure 1.1: Example of Multicast Islands

possible solution to this problem is to connect multicast islands using Application Layer Multicast. By doing so, the benefits of both techniques can be gained. This technique is referred to as Hybrid Multicasting.

This thesis discusses our novel approach to hybrid multicasting which utilise AMT tunnels and ALM together with a Peer-to-Peer overlay network to connect native multicast islands. We termed our technique Opportunistic Native Multicast (ONM). The thesis demonstrates the performance of the approach through simulation considering different levels of node churn. Furthermore, in order to further improve the performance of the approach, besides the basic configuration with a Primary Node (PN) only, a single or multiple additional Secondary Nodes



can be used as backup. An election algorithm to identify these nodes and present results for ONM for different heartbeat message intervals are introduced and used. Finally, the thesis makes recommendations regarding the most suitable configuration of ONM considering different network conditions.

## 1.2 RESEARCH PROBLEM

Currently, the Internet is fragmented into many isolated areas that are not connected using unicast-only connections due to the lack of support of native multicast in the backbone and because different types/protocols for native multicast are deployed. This is a major issue for content distribution as multicast requires the support of intermediate routers [8]. Since multicast packets are dropped when leaving a Multicast Island and the backbone of the internet does not support multicast [9], Native Multicast (NM) cannot be used as an Internet-wide to distribute data. Alternatively, Application Layer Multicast (ALM) can be used. However, this is not as efficient as Native Multicast (NM) [5]. A solution to the problem of Multicast Islands is to connect these Multicast Islands using ALM [10]. This approach is termed Hybrid Multicast (HM). By doing so, the benefits of both techniques can be exploited.

Currently, there exist different approaches to connect multicast islands. However, they suffer from different shortcomings, such as, the need for manual configuration, lack of fault tolerance and recovery, lack of standardisation or incompleteness. ONM, by using ALM for the overlay and AMT to tunnel multicast between islands, solves many of these issues. In this thesis, the chosen ALM and Peer To Peer (P2P) algorithms are mature and widely used, such as in FreePastry [11] for Pastry, JANUS [12] which implements SCRIBE.

### 1.3 THESIS STATEMENT

To solve the issues discussed in 1.2, this thesis propose a Hybrid Multicast (HM) system to connect these islands utilising Automatic Multicast Tunnelling (AMT) and Application Layer Multicast (ALM). The system takes advantage, for the first time, of AMT to tunnel traffic between islands. While implementing Hybrid Multicast (HM) introduces a number of challenges, the proposed system tries to address them. Some of the challenges that are introduced and addressed in this thesis: availability, interoperability, self-adaptation and efficiency. In our proposed model, called Opportunistic Native Multicast (ONM), nodes belonging to the same island discover each other and join the same native multicast tree. Using native multicast capability of the island improves the efficiency. Also, it allows nodes to seamlessly join the multicast group as the islands are connected using AMT.

### 1.4 AIMS AND OBJECTIVES

The support for unicast only in the Internet backbone and the support for multiple NM approaches resulted in multicast-capable parts of the Internet being scattered across the network as islands. The aim of this thesis is analyse the issue of multicast islands. The thesis will propose a system that connects these islands while utilising the islands' local capabilities, which can often be more efficient and robust. Also, this thesis will evaluate the possibility, applicability and the performance of the proposed system. It is hypothesised that using ONM will provide a significant improvement over using conventional multicasting methods.

The objectives of the thesis are:

- Due to the issue of multicast islands discussed in the Thesis Statement section above, this thesis introduces a method to connect different multicast islands using AMT tunnels while allowing the islands to utilise their native multicast capability.

- Design and implement how ONM builds multicast trees while maintaining efficiency and reliability of delivery. Furthermore, the approach will cope with different levels of node churn in the network.
- The proposed system allows for one node per island to be responsible for the delivery and the connectivity between the Native Multicast (NM) tree and the Application Layer Multicast (ALM) tree. This node is referred to as the Primary Node (PN). Due to its importance, the thesis investigates how the election process of the Primary Node (PN) takes place.
- Also, investigate how the islands can detect when the Primary Node (PN) is not available any more. Also, how can the island react to this event and select another Primary Node (PN). This is important since the Primary Node (PN) is susceptible to failure and disconnects and it might get churn out of the network.
- Design a way to improve efficiency and recovery speed from Primary Node (PN) failure by allowing the Primary Node (PN) to select a backup node that takes over when the Primary Node (PN) is not available anymore. This node is referred to as the Secondary Node (SN).
- Investigate the frequency of the exchanged heartbeat messages and find the optimum frequency of communication between nodes, between nodes and the Primary Node (PN), and between the Primary Node (PN) and the Secondary Node (SN).
- Allow islands to detect the stability of the network and change the frequency of the heartbeats accordingly.

## 1.5 CONTRIBUTIONS

The contributions of this thesis are highlighted in the following list:

- To understand different approaches of multicasting, a review of different multicast techniques and approaches were identified and presented. The

review has identified existing solutions of multicast and focused on the ones that have attempted to bridge different islands. Also, related technologies that were used such as P2P and AMT were identified.

- Due to the lack of the support of the current simulators of hybrid multicast and AMT, an extension was built to the simulator used in the research. Using this adaptation, the thesis implemented and compared network environment with varying values of island size and network capabilities.
- A novel Opportunistic Native Multicast (ONM) approach was designed. The proposed system aims to connect islands and utilising local native capabilities of the network. The synergies between Application Layer and Network Layer are maximised by developing native multicast detection and island awareness techniques.
- Develop, analyse and experiment with the process to elect a Primary Node (PN). The process needs to allow nodes to interact with other nodes in their islands. Nodes should be able to detect the need for an election and unanimously agree on the next Primary Node (PN) at the end of the election.
- Investigate the use of backup nodes, Secondary Node (SN), to improve the availability and robustness of Opportunistic Native Multicast (ONM). Also, the thesis investigates the cost and the improvement achieved by introducing multiple SNs.
- The thesis introduces a number of mechanism for selecting Primary Node (PN) and the Secondary Node (SN). This allows ONM to select nodes that are more stable which in turn, improves the overall stability of the network.
- The thesis evaluates the performance of ONM under a number of real life network configuration and parameters. This ensures that the system is deployable in practice and improves performance when compared to different ALM systems.

- Opportunistic Native Multicast (ONM) significantly improves on the performance of previous approaches in terms of delivery of messages and overhead incurred. In this thesis, one of the major Hybrid Multicast (HM) systems is compared to Opportunistic Native Multicast (ONM).

## 1.6 THESIS SCOPE AND STRUCTURE

This thesis will define the main issues with using Native Multicast (NM) and Application Layer Multicast (ALM). These issues are mainly: Multicast Islands and taking advantage of network capability to increase efficiency. After that, the currently proposed Hybrid Multicast (HM) techniques will be discussed. Then, the thesis will propose a Hybrid Multicast (HM) approach to solve these issues.

The thesis is structured into the following chapters:

- Chapter 1, **INTRODUCTION**: This chapter introduces the thesis research objectives and aims. Also, it identifies the research questions that the thesis is trying to solve.
- Chapter 2, **BACKGROUND AND RELATED WORK**: This chapter discusses P2P and its advantages. Moreover, it will have a look at different types of P2P protocols and how to use them to achieve ALM. It reviews multicast and its protocols and discusses different ways to achieve multicast and compare their advantages and disadvantages.
- Chapter 3, **OPPORTUNISTIC NATIVE MULTICAST**: This chapter discusses our proposed method to achieve hybrid multicasting. It discusses fail recovery and redundancy mechanism.
- Chapter 4, **PERFORMANCE EVALUATION OF ONM**: This chapter evaluates our proposed protocol and compare its effectiveness under different settings. It also, fine tunes different values for variables used in the implementation to find what yields the best results.

- Chapter 5, CONCLUSION AND FUTURE WORK: This chapter, based on the results collected in chapter 4, draws conclusions and suggest ways to take this work forward.

## BACKGROUND AND RELATED WORK

---

This chapter discusses the background of the three main related technologies: Peer To Peer (P2P), Multicast and Automatic Multicast Tunnelling (AMT). For P2P, an overview will be given of its different types: Structured and Unstructured. Then, this chapter will discuss some of the key examples of each type. Secondly, the Multicast section will discuss the different approaches of how a multicast tree can be built and deployed. Examples protocol for each approach will be discussed. Lastly, for Automatic Multicast Tunnelling (AMT), the chapter will provide an overview on the operation and approaches that can be used to build tunnels.

### 2.1 MULTICAST

In computer networking, multicasting is the simultaneous delivery of a message to a group of destination computers. Some important Internet applications, such as VoIP, video streaming and recently Augmented Reality (AR) and Virtual Reality (VR), rely on or take advantage of Multicast capabilities of the network. However, one of the challenges faced by AR and VR is the lack of Native Multicast support [13]. Moreover, issues with native multicast has affected the design of protocols.

Issues	IP Multicast	ALM
Multicast efficiency (delay/bandwidth)	High	Low - Medium
Complexity or Overhead	Low	Medium - High
Ease of deployment	Low	Medium - High
The OSI layer	Network Layer	Application Layer

Table 2.1: Conceptual comparison of IP multicast and ALM

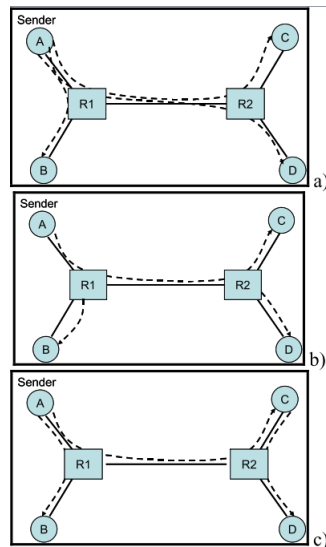


Figure 2.1: a) Unicast, b) IP Multicast, c) Application Layer Multicast.[1]

For example, in designing MP-ALM, ALM was chosen over Native Multicast due to these issues [14]. In Internet of Things (IoT) domain, multicast plays an important role on efficiency and power consumption [15]. Also, many of video streaming protocol did not utilise Native Multicast due to issues with respect to address allocation, routing, authorisation, group management, security, and scalability [16].

Since routing is carried out by Layer 3 routers in the Internet, multicasting was designed, initially, to operate in Layer 3 in the Open Systems Interconnection (OSI) network stack [17]. In this case, it is called Native Multicast (NM). However, different methods can be used if Native Multicasting is not supported throughout the whole network. One of these methods is Application Layer Multicast (ALM). In ALM, Layer 7 of the OSI network stack is used instead of Layer 3 [18]. Finally, we could use a hybrid approach employing ALM and NM where both layers (i.e. 3 and 7) are used. This method is called Hybrid Multicast (HM).

As shown in Table 2.1, these types of multicasting can be classified by the OSI layer they operate in. We can see from the table that Native Multicast is more efficient and less complex than ALM. On the other hand, ALM is easy to deploy without the need for the support of the network infrastructure [19].



Figure 2.1 illustrates the different methods that messages can be sent in the network. The figure illustrates three cases: a, b and c which corresponds to Unicast, Native Multicast and ALM respectively. In the figure, Node A is the source of the data which will be sent to Nodes B, C and D. In case a in Figure 2.1, the source sends three different unicast messages - to each receiving node. The routers forward the messages to each receiving node. In case b, Node A sends a single copy of the message. The routers will deal with forwarding and duplicating the message to each receiver. Router R1 will have Node B registered as part of the multicast group so it will send a copy to Node B. Since R2 has Nodes C and D registered, R1 will send another copy to R2. Finally, R2 will send copies of the message to Nodes C and D. Lastly, in case c Nodes A, B, C and D will form a multicast tree. The routers will just forward the exchanged messages without taking part in the multicast decisions. Node A represents the head of the tree with Nodes B and C acting as leaves. Node D is a leaf to Node C. So, when Node A tries to multicast a message to the group, it will send it to all the leaf nodes connected to it, i.e. B and C. Then Node C will forward the message to Node D.

Each of the multicast approaches will be discussed further in this chapter. Section 2.1.2 discusses NM in more detail, followed by section 2.1.3 discussing ALM, and finally Section 2.1.4 looking at a hybrid approach that combines Native Multicast and ALM.

### 2.1.1 *Taxonomy*

There exists different methods for classifying Multicasting Approaches. Multicast Protocols can be classified by the routing algorithm used by the protocol to construct the acyclic spanning tree. The routing protocol has two main responsibilities: to build and manage the state information and to select the most appropriate path to carry information on [20]. Also, Multicast protocols can have additional responsibilities such as group management and QoS. Moreover, to increase the efficiency, Multicast algorithms try to build Minimal Spanning Tree

(MST). These protocols can be classified by the type of tree that is built by the routing algorithm [21] as following:

- **Source Tree:** Source Tree Algorithms, or Shortest Path Trees, build a separate tree for each source. To connect a receiver to a source, the protocol uses Reverse Shortest Path (RSP). This path is built using Reverse Path Forwarding (RPF) at the intermediate nodes. This Approach is very efficient for high data rate sources as data always traverse the shortest path to the receiver. However, this increase the cost as the number of groups and the number of senders per group increases [22]. An example of Source Tree algorithm is Multicast Extension To Open Shortest Path First Protocol (MOSPF) [23] which use Dijkstra's algorithm to build the Shortest Path Tree. Another example is PIM Source-Specific Multicast (PIM-SSM) [24].
- **Shared Tree:** Here, a single tree is built to be used by all sources. The flow of data in the tree can be unidirectional or bidirectional. This approach is efficient for cases where sources send a low rate traffic. However, it increases the traffic concentration. In Shared Tree algorithms, a single location in the network is called a Rendezvous Point (RP). At the RP, all data from the sources are sent to all receivers. Compared to Source Tree, an increase in the delay may occur as data sent to the RP then to all nodes may not follow the shortest path. Additionally, the selection of a node to act as RP is critical to the performance of Shared Tree algorithms. An example of Shared Tree is found in CBT [25], SCRIBE [26] and PIM-SM [27].
- **Steiner Tree:** Steiner Tree uses other information to build the spanning tree by giving links weights or costs [28]. Steiner Tree minimise the the total cost of a shared tree at the expense of delay. Finding such tree is a NP-complete problem. So, approximation algorithms have been proposed such as Kou, Markowsky and Berman (KMB) [29]. The cost resulted using KMB averages 5% more than the Steiner Tree. However, KMB requires full network topology to build the tree so it is not practical for large networks. To overcome the issue of the delay, especially for delay sensitive application

such as VoIP, a variant of Steiner Tree that is delay bounded is proposed. Delay Bounded Steiner Trees [30] will build a tree with minimum cost but under a set delay limit.

- **Reduced Tree:** As a way to solve the scalability issues with Multicasting, Reduced Trees were proposed. Here, the tree does not have any relay nodes (of degree 2) [31]. This will result in about 80% reduction in the amount of states maintained for each multicast group.
- **Incremental Distributed Asynchronous Algorithm for MST:** To avoid the need for recomputing the Minimal Spanning Tree (MST) from scratch with every change in the topology, it proposes using the existing tree and updating it asynchronously. It also requires the knowledge about adjacent edges only [32].
- **Bounded Shortest Multicast Algorithm (BSMA):** First, Bounded Shortest Multicast Algorithm (BSMA) [33] constructs a tree with the minimum delay at a given source that will span all the group's members. Then, it will iteratively replace high-cost edges with less costly ones while keeping under the delay constraint. This will repeat until no further links can be replaced without exceeding the delay constraints.
- **Bauer Algorithm:** Using the algorithm proposed by Bauer [34], constraints are imposed on the number of outgoing links for each group. At the beginning of the tree construction, links are added one at a time to a random starting point. This is repeated for different starting points. A rearrangement is triggered when a threshold of damage index as nodes join or leave.
- **Delay Variation Multicast Algorithm (DVMA):** Here, the delay and delay variation, or jitter, is considered when building the tree. The jitter is overcome by using buffers at the source, intermediate and receiving nodes [35]. Also, it will cost more as additional information is needed at different nodes at the network. However, it will result in a better variation and more efficient buffering. A Delay Variation-Bounded Multicast Tree (DVBMT) is

used to build a tree that is bounded by delay and delay variation. DVBT starts with a tree that satisfies the constraints which may not include all nodes. Then, the algorithm tries to insert nodes in the network while not violating the set constraints.

- **ARIES / GREEDY / Edge Bounded Algorithm (EBA):** Multiple algorithms were proposed to deal with dynamically updating large network of point-to-point networks, e.g. A Rearrangeable Inexpensive Edge-based online Steiner Algorithm (ARIES) [36], GREEDY [37] and Edge Bounded Algorithm (EBA) [38]. For a new node to join ARIES, it will be assigned to an existing node in the tree using Geographic-Spread Dynamic Multicast (GSDM) to choose one with the greatest geographical spread. GREEDY aims to minimise the spread of the tree as much as possible. A new node will connect to the closest node tree node using the shortest path. EBA create a limit on the distance between nodes in the tree for each change. A rearrangement will take place in EBA when the distance exceeds a set limit.

Another dimension to classify multicast is using what layer of the OSI stack does the tree multicasting occur. This is more relevant to our work since we are trying to combine different layers of the OSI stack to achieve multicasting globally. Therefore, we will use this dimension when discussing different multicast algorithms. As can be seen in Figure 2.2, there exists three types: Native Multicast (NM), Application Layer Multicast (ALM) and Hybrid Multicast (HM). In Native Multicast (NM), the multicast is achieved by the routers in the network fabric. Alternatively Application Layer Multicast (ALM), the multicast operations are handled by the end systems without requiring support from the network routers. In Hybrid Multicast (HM), these operations are handled by both layers. In this section, these will be discussed.

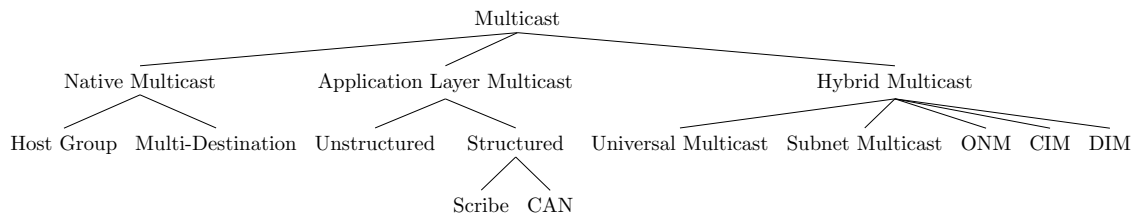


Figure 2.2: Multicast Taxonomy

### 2.1.2 Native Multicast

In Native Multicasting NM, the forwarding and replication of the multicast packet is carried out at Layer 3. This means that the packet sent to a group is sent only once by the source node. The network infrastructure will handle replicating the packet as needed until it reaches every destination in the multicast group. There exists different approaches to implement NM such as:

- **Host Group Multicast (HGM):** In this approach, the maintenance of groups' hosts is handled in the routers.
- **Multi-Destination Routing (MDR):** Here, the sources are responsible for the maintenance of their groups' hosts [39].

Native multicast, such as MDR and HGM, reduces the number of messages in the network dramatically [40]. However, these approaches rely on their widespread deployment, which is slow and often encounters a number of issues; For example, the network infrastructure must be configured to support the same multicast protocol. If any region of the network does not support native multicast or support a different native multicast protocol, messages might not get forwarded and passed in this region. This leads to multicast islands.

#### 2.1.2.1 Host Group Multicast

In Host Group Multicast, Layer 3 routers implement the multicasting, that is routers manage and maintain groups. HGM creates a group address per multicast tree with the routers keeping track of the active group addresses [1]. Figure 2.3 shows an example of the operation of HGM. HGM works best for a relatively

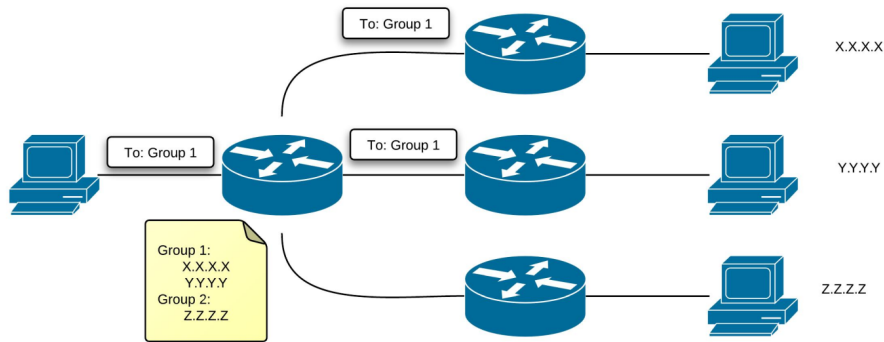


Figure 2.3: HGM Multicasting

small number of groups that can have a large number of subscribers. As more groups exist, more memory is needed in the routers. Clearly, this limits the number of groups that a network can support. Since the average lifespan of a session is relatively short, the states in the routers will need to be updated frequently. Furthermore, the overhead of a node joining a group is considerable as this information needs to converge in the groups tree. The most common protocol used with HGM in IPv4 is IGMP.

The Internet Group Management Protocol (IGMP) is a protocol that is used between hosts and adjacent routers to establish and manage multicast groups. The protocol keeps track of the available multicast groups and makes sure that the group messages are forwarded consistently without any unnecessary duplication. This is accomplished through the exchange of IGMP messages:

- **Membership query:** This is sent by routers to ask hosts if they want to be member of a multicast group.
- **Membership Report:** This is sent by hosts to let the router know that they are interesting in joining a multicast group.
- **Leave Group:** This messages is sent by hosts when they are no longer interested in receiving the multicast traffic.

The type of the messages is indicated in the type field in the IGMP message(2.4).

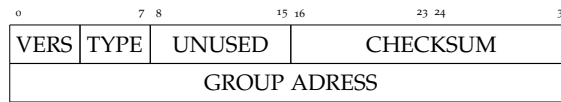


Figure 2.4: IGMP Message

### 2.1.2.2 Multi-Destination Routing

In Multi-Destination Routing (MDR), each multicast packet is sent with a list of its destinations. Enabled routers will route this message and keep it as one message as long as all the destinations have the same next-hop. Otherwise, the router will split the addresses and duplicate the message, routing each one with its corresponding list of destinations, to their own next hop. This is repeated until a message has only one address as its destination. At this point, the message becomes a unicast message and is routing accordingly. See Figure 2.5.

The advantage of MDR is that it does not require routers to keep states of active multicast groups thus making it highly scalable allowing it to support a high number of groups. This has made MDR suitable for relatively small-sized groups that have a short life span. In that case, network routers do not incur overhead when there is a change in the group e.g. create or delete a group. Another advantage is that there is no overhead in the network when hosts join or leave groups.

On the other hand, MDR uses a special type of packets that supports multiple entries as destinations. This requires that routers support this extension to be able to route the message correctly. Also, MDR adds processing overhead at each router when routing multicast messages as the routers may need to split packets. Moreover, the sender is responsible for managing the multicasting group. Also, there is a limitation on the group size due to the fact that all of the destination addresses must fit in the header of the message.

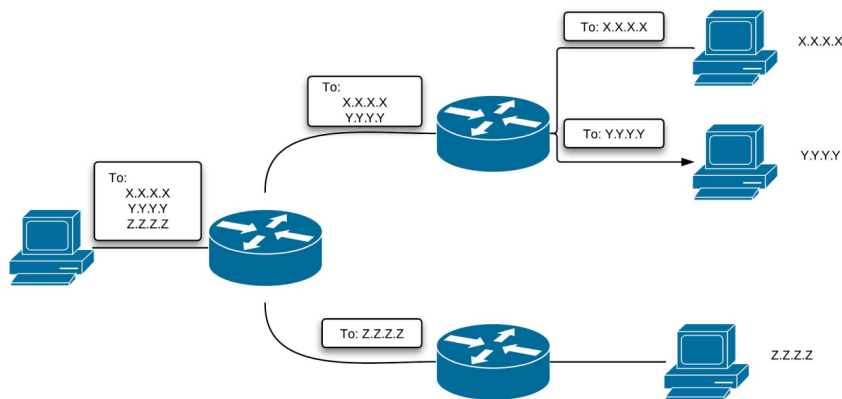


Figure 2.5: MDR Multicasting

### 2.1.3 Application Layer Multicast

An alternative to Native Multicast is Application Layer Multicast ALM. With ALM, the forwarding and replication is carried out in Layer 7 of the OSI model. There are different approaches that can be used to achieve ALM. These approaches can be classified into two groups: mesh-first, and tree-first. Mesh-first works by building a richly connected graph first and then construct the multicast tree. However, the tree-first approach creates the distribution tree first and then subsequently adds additional links for control. Using the mesh-first approach allows to build more than one multicast tree using the same overlay mesh. Since our approach aims at connecting islands which can carry multiple Multicast Trees, the mesh-first approach can be achieved. The optimum result happens when the design of the ALM Layer 7 tree is based on the IP Layer 3 topology. However, this is difficult with structured overlays with a predefined overlay topology regardless of the physical location of nodes. Since the overlay network connections are abstract links with no low-level details, there might be a mismatch between the overlay structure and the physical network topology. For example, two nodes can be physically located approximate too each other. However, in the overlay graph, they are separated by several nodes. Despite these issues, ALM does not require infrastructure support which makes it easy to deploy. Since there is no support in



most part of the internet for Native Multicast protocols, ALM becomes the only way to carry multicast traffic.

The performance of ALM is measured by the following factors:

- **Stretch:** is the delay of the overlay path over the delay of a unicast message.
- **Stress:** is the number of identical copies of a message carried by a link or a node
- **Control Overhead:** is the number of control messages needed to make the ALM work.

#### 2.1.3.1 CAN-Multicast

Content Addressable Network CAN is a distributed infrastructure that provide a hash table functionality that map keys to values [41]. CAN is designed to not only be used to share files across the peer-to-peer network but as system to efficiently distribute content in an Internet-like scale. CAN-Multicast was designed as CAN extension to provide ALM on top of CAN [42].

CAN-Multicast uses CAN overlay to distribute multicast messages across the multicast group. If all of the peers in the overlay belong to the same multicast group, flooding is used to distribute content across the CAN overlay. Otherwise, multiple mini-CAN systems are created; one for each multicast group. In the later, the underlying CAN system serve as a base to create the needed mini-CAN systems. To create the mini-CAN systems, a multicast group is mapped to a node in the underlying CAN system. This node will act as a *bootstrap* in the creating of the mini-CAN.

To propagate multicast messages in the network, two methods were proposed:

**NAIVE FLOODING** When the multicast message reaches a peer, the peer will check if it had already received the message. If the message is new, the peer will flood the message to all of its neighbours except for the one that it came from. This method is simple but it will produce a large message replication.

**EFFICIENT FLOODING** Each node will analyse its location and its routing table to decide which node to forward the message to. The source of the message will flood it to all of its neighbours. However, subsequent forwarding will depend on its relationship to the sender. When the message reaches a peer in the network from a source in the dimension  $i$ , it will forward it to peers who are further on the same dimension  $i$ . Also, it will forward it to peers on the other side of the dimension  $i$ .

#### 2.1.3.2 *SCRIBE*

Scribe is a large scale and decentralised application level multicast infrastructure [26]. Scribe uses Pastry as its underlying peer-to-peer network.

In Scribe, the multicast address is generated by hashing the creator address and the group name. The generated address is considered a unique ID that can be used in the Pastry overlay. Then, the creator will send a create message to the node in the pastry overlay responsible for the generated address which would be the node with the closest ID. This node is the rendezvous point for this group. Moreover, for decentralisation purpose, all nodes that forward the multicasting traffic will snoop in on subscribe messages passing them. If the forwarding node happens to subscribe to the same multicasting group, it will not forward it to the root node and will act as a parent for the subscribing node. This behaviour will result in a tree structure for each multicast group. Then, a series of keep-alive messages would be sent periodically from a child to its parent. When the parent is unreachable, the child would send a new join message to the root node. Also, keep-alive messages would be sent from the parent to each child. This will allow the parent to detect any child that is no longer reachable. As for the root node fault-tolerance, Pastry will handle such failure since keys in the network are duplicated across multiple nodes.

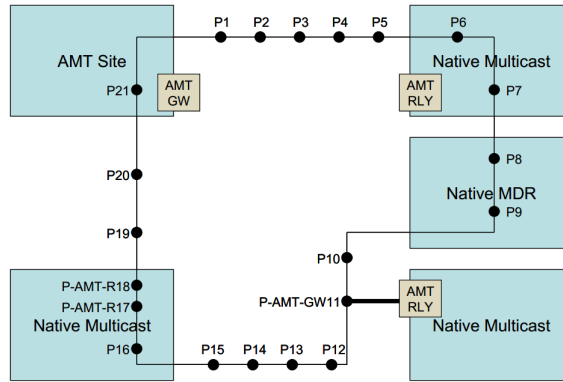


Figure 2.6: A Sample Hybrid Multicast Network. [2]

#### 2.1.4 Hybrid Multicast

Each discussed approach, i.e. Native Multicast and Application Layer Multicast, has its own advantages and trade-offs. However, Hybrid Multicasting tries to combine the two approaches to give better performance. In Hybrid Multicasting, Layer 7 is used to perform multicast operations e.g. tree creation, joining, leaving, and re-forming a tree. Wherever possible, tree operations are mapped to native multicast [43].

In Figure 2.6, we have a hybrid multicast network from [2]. In this network we have four Native Multicast domains. MDR is used in P8 and P9. While P21 joins the multicast group via an AMT Gateway (AMT-GW). Any peers that reside outside any IP (layer 3) multicast domain, marked in Blue squares, will use ALM (layer 7). Moreover, a relay is used to connect any peers that do not have native connectivity to the multicast backbone. These relays could be another peer. While some peers could act as a gateway and connect to a relay.

One of the earliest attempts to connecting multiple native multicast islands was Mbone [9]. Mbone was an experimental backbone to carry multicast packets across the internet that requires a specialised hardware and software. Mbone related approaches usually required an administrator to set up and maintain tunnels. Moreover, Mbone suffered from some issues of access control, security, address allocation and network management[6]. Automatic Multicast Tunnelling (AMT) was proposed to automate the management of tunnels [44]. AMT defines

specific devices, Relays and Gateways, in the network to interact with the native multicast protocol and to establish tunnels when needed. As such AMT focuses on connecting island pairs rather than a global unified network. AMT will be discussed in more detail in Section 2.3.

Except for AMT, one common limitation of the listed approaches is that Layer 4 information are encapsulated and hidden. Layer 4 information are needed to do Quality Of Service (QoS) and traffic shaping. QoS is very important when the multicast carries a real time data [45] e.g. VoIP and video conferencing.

Tunnelling multicast over unicast-only network is an important aspect of Hybrid Multicast. These tunnels play a vital rule in the performance of multicasting. We can classify these techniques by the way these tunnels are built into two types:

**EXPLICIT TUNNELLING** With Explicit Tunnelling, isolated multicast islands are interconnected using manual tunnels such as Generic Routing Encapsulation (GRE). This can be done by the network administrators manually where each tunnel is designed and maintained. In the case of Multicast over GRE, each multicast packet get encapsulated inside a GRE header[46]. This encapsulation results in that the layer 4 information is hidden to the router connecting the two end point of the tunnel. This information is needed for doing some QoS and traffic shaping.

**WITHOUT EXPLICIT TUNNELLING** Here, tunnels will be created and maintained automatically. The shape and topology of these tunnels will change dynamically as the underlying network changes. This could be done using different techniques such as peer-to-peer overlays. We can categorise these protocols into ALM-First (e.g. HM [43], NICE [19]) and Native Multicast-First (e.g. HIPM [47], ASRM [48], UM [49]).

#### 2.1.4.1 *Automatic Multicast Tunnelling AMT*

Automatic Multicast Tunnelling AMT provides a way to build the unicast tunnels between multicast islands when needed without explicit configuration. The multicast tunnels will keep the layer 4 information in the packet allowing for

better QoS. More detail look on AMT operation is in section 2.3. However, AMT lacks the peer to peer overlay to manage for its scalability. [2] attempts to solve this issue by extending RELOAD to work with AMT.

#### 2.1.4.2 *Island Multicast*

In Island Multicast (IM), Native Multicast Islands are combined and connected using overlay data distribution [50]. Island Multicast (IM) presents two different protocols to solve the problem of Island Multicast:

**CENTRALISED ISLAND MULTICAST (CIM)** This protocol is suitable for groups with small size, have many-to-many communication and high bandwidth requirement e.g. multi-party conference calls. Here, there is a central server to build and maintain the spanning tree. While this allows for fast building of small multicast groups, it does not scale well for a high number of participants. Also, the need for central servers in Centralised Island Multicast (CIM) has created a resource bottleneck and a single point of failure.

**DISTRIBUTED ISLAND MULTICAST (DIM)** This protocol is suitable for large groups where scalability is required. In Distributed Island Multicast (DIM), hosts in the same multicast island elect a unique leader. The leader node will be responsible for constructing the delivery overlay. So, DIM will form a two-tier multicast trees where the island leaders form an overlay tree. Also, each pairs of island leaders will form a bridge between them. This increases the overhead of the protocol significantly as will be seen in later sections. Distributed Island Multicast (DIM) is very comparable to the thesis proposed system, Opportunistic Native Multicast (ONM) and will be used as a benchmark for the performance of ONM.

#### 2.1.4.3 *Universal Multicast*

As been proposed in [51], Universal Multicast UM provides a way to connects multicast islands using dynamically build unicast tunnels. UM allows for multiple

connections for an island. Doing so will allow for an improvement in speed for large islands. Inside each multicast island, one or more Dedicated Members DM are elected to natively deliver the multicast to the island using Native Multicast. However, the DM must advertise for its existence to all other nodes in the island periodically. The frequency between advertisements determines how fast can an islands converge and elect a new DM. There are no backup nodes that detect the failure of the DM that communicate more frequently and hence will react faster to DM failure.

Also, the authors of Universal Multicast have proposed a protocol for intra-island multiple-DM management protocol [43]. This protocol is called Host Group Management Protocol HGMP. HGMP is concerned with dynamically electing peers to be Dedicated Member and using multiple Dedicated Members for load-balancing purposes.

However, Universal Multicast (UM) requires that the joining host to search the Native Multicast (NM) one hop at a time. This has rendered Universal Multicast (UM) to suffer from a long joining delay [52] especially for a large group sizes. Also, Universal Multicast (UM) cannot handle a dynamically changing network [52].

#### 2.1.4.4 *Multicast Delivery Based on Unicast and Subnet Multicast*

In [53], the paper proposes Subnet Multicast (SM) for connecting multicast islands by having the source island forward a copy of the data to each receiving island. The proposed protocol does not depends on the native multicast protocol. However, there still some issues of scalability and its incompleteness compared to the other proposed solutions. SM requires the source island to maintain a database with entries for all other islands. Also, SM assumes that the data is sent as a unidirectional stream in an one-to-many relationship.

#### 2.1.4.5 *Hybrid Multicast Issues*

Hybrid Multicast is a promising solution to deploy multicast globally across the Internet while utilising the native support in part of the network. However, there

exists a lot of challenges and issues that need to be considered. [8] deals with some of the issues with Island Multicast. The issues can be classified into the following:

**NAT** Due to its operation across multiple different networks, the problem of NAT devices will affect how peers discover each other. This issues originates in the peer-to-peer paradigm. Consequently solutions proposed for P2P also apply to Hybrid Multicast. Solutions proposed in [54],[55] are typical examples.

**FAULT TOLERANCE AND RECOVERY** Due to the nature of the Internet, losses and failure are expected. This issue can be more critical since peer-to-peer systems will need time to converge and recover from any change to the topology. While some methods have been used to negate the effect of losses, some dramatic changes to the underlay or the peer-to-peer overlay can be drastic. Moreover, some multicast application can be sensitive to delay and packet-loss such as Voice Over Internet Protocol (VoIP) and Internet Protocol TeleVision (IPTV). Also, multicast has been used in crisis management systems which relies heavily on fault tolerance and recovery [56].

**STANDARDISATION** Due to the different ways protocols can be coupled and combined, it is very challenging to design a standard that is optimised for all of these scenarios. Also, since the Multicast Islands may belong to different administrative entities, a unified standard that is agnostic to the underling topology is preferable.

**SECURITY** As in every network, the multicast network is susceptible to different types of attacks. Moreover, as the network combines different types of protocols, it will inherit some of their weakness. Especially, the existence of malicious nodes and peers poses an issue albeit in a very similar way as in peer-to-peer overlays and native multicast. Consequently, existing techniques from these two domains can also be applied to Hybrid Multicast. Some proposed mitigations can be found in [57], [58].

## 2.2 PEER TO PEER

Peer to Peer (P2P) networks are a decentralised and distributed type of network where every node assumes client and some server functionality [59, 60, 61].

The peers will join to a virtual overlay that is 'overlaid' over the physical connections of the network. Those nodes or peers will collectively provide different services to the network such as storage and routing. Using the overlay, nodes can query other nodes and services. Then, an out of bound connection can be used to complete the request.

By using P2P overly, node can join and leave with minimum friction. With the use of a peer-to-peer network, instead of using the centralised Server-Client model, some wanted features are introduced, such as:

- **Decentralisation:** With no single point of failure, the peer-to-peer network is more robust and balanced.
- **Self-Organising:** As peers join or leave the network, the topology of the peer-to-peer network is adjusted.
- **Scalability:** The size of the peer-to-peer network can vary dramatically over the network lifetime. So, scalability is important to support this growth without putting stress on a single peer or the network as a whole.
- **Availability:** With changes in the underlay network, some paths or nodes can fail. Peer-to-peer networks are designed to recover from these failures.

Broadly, P2P can be classified into: Centralised, Decentralised and Hybrid P2P system [59]. Centralised P2P system uses dedicated server to maintain the overlay. A famous example of these systems is Napster [62]. These servers will assists in locating other nodes and services as they will have a look up tables. However, they act as a single point of failure and suffer from scalability issues as the number of nodes increasing rendering the servers a bottleneck. Thus defeating the purpose that this thesis is using P2P for.



Hybrid P2P systems allow for some nodes in the overlay to act as super peers. These super peer will have responsibility similar to Centralised P2P systems but they will be dynamically allocated improving the availability of the overlay. Examples of these overlays are Gnutella 0.6 [63], Gia [64] and JXTA [65]

In Decentralised P2P systems, all nodes play similar roles and share the same responsibility. This will make the system more robust as it does not have a single point of failure. The Decentralised P2P system constitutes the majority of the protocols used and is the focus of our thesis as it will be used at with our system.

According to the structure or the lack thereof, Decentralised P2P systems can be further classified into: Structured, Unstructured and Hybrid overlay. In this section, we will discuss each type in more details.

### 2.2.1 *Taxonomy*

Commonly, Peer to Peer algorithms are grouped into Structured and Unstructured overlays. In structured peer-to-peer overlays, there is an overall structure that is imposed on the network. Based on this structure, nodes in the peer-to-peer network form links and locate resources. On the other hand, unstructured peer-to-peer nodes form links arbitrarily. This makes the structure of the network hard to know and control.

In the unstructured model, there is no structure that is imposed on the overlay. Instead, nodes connect to each other randomly or in an ad-hoc manner. Due to the lack of structure, the network is easy to build and can easily adjust to a high churn rate. However, there are some severe limitations of the design of the unstructured peer-to-peer overlays. One main disadvantage is that the search algorithm is not deterministic. In the structured peer-to-peer algorithms, searches are deterministic and the average latency to complete search requests is known in advance. However, unstructured peer-to-peer search algorithms does not offer any guarantee that the searched node can be located. Also, the search process may visit the same node more than once [66].

### 2.2.2 Structured Peer-to-Peer Overlay

Having a structure imposed on the network can improve the search efficiency and guarantee that the resources will be reached within a small number of hops [67, 68]. In structured P2P, resources are mapped to peers in the overlay. Commonly, this mapping is achieved using a hash function of the resource data or name. So, when querying a resource, such as a file name, a peer will have a direct its look up to a particular node in the overlay responsible of that resource.

In structured P2P networks, peers use Distributed Hash Table (DHT) algorithms to store and locate peers and resources. In DHTs, peers are assigned a unique ID which is typically generated by hashing the address of the peer. Also, IDs for resources, such as files and services, are generated by hashing the name of the resource. Resource IDs are stored on nodes whose hash is closest to the resource ID plus possibly on some neighbouring nodes for redundancy.

In structured peer-to-peer overlays, the network rely on the DHT to manage the structure and the operation of the network. DHTs provide a list of characteristics for peer-to-peer and key advantages for multicasting. These advantages are:

- **Decentralisation:** With no central point of failure, the DHT provide robustness to the network.
- **Fault Tolerance:** DHT systems can be reliable despite changes in the topology.
- **Scalability:** With the size of the network increasing, DHT adapt and maintain their performance.

Other properties can be added to the system as required, for instance by implementing security and anonymity on top of the DHT.

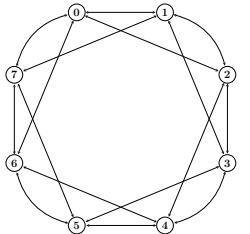
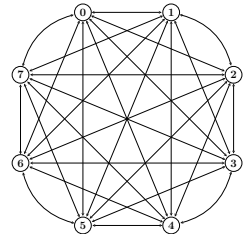
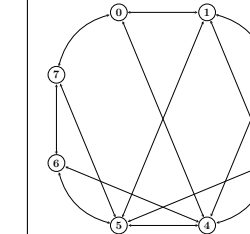
	Structured		
	Multi-Hop	One Hop	Variable Hop
Connectivity:	Symmetric	Fully Meshed	Asymmetric
Examples:	Chord, Pastry	Epichord, OneHop	Accordian
Sample Structure:			

Table 2.2: Comparing Different Types of Structured Peer-To-Peer Networks

### 2.2.3 Types of Structured Overlays

A structured peer to peer network can be classified by the upper bound of hops for a request to reach its destination as seen in Table 2.2. According to this classification, there exists three main types:

#### 2.2.3.1 Multi-Hop

Here, each node in the overlay connects to a subset of the total number of nodes. Since the resulting structure of the graph is symmetric, it will have an upper bound on the number of hops needed to find a resource. Since a node must only keep track of a subset of the nodes, the routing table is reduced. By reducing the size of the routing table, it will reduce the overhead needed to update it. For example, Pastry [69] will take a maximum of  $O(\log_2 n)$  hops. Other common examples of Multi-Hop overlays are: Chord [70] and Kademlia [71].

**PASTRY** A Pastry overlay [69] has a circular structure and each node is assigned a 128-bit ID. Based on the modulo  $2^{128}$  of the ID, the nodes are ordered in the structure and its will know its place within the circular address space. Each node keeps routing tables and tries to forward requests to the closest nodes. This

Rows	0	1	2	3
0	02030213	↓	22030213	32030213
1	10231012	11023121	↓	13232103
2	↓	12120301	12232231	12303221
3	12003210	12012233	12021002	↓
4	↓	12031231	12032120	12033001
5	12030032	12030112	↓	12030322
6	12030201	↓	12030223	12030230
7	12030210	12030211	12030212	⊙

Table 2.3: Example of Pastry Routing Table for a node with ID 12030213 and  $b = 2$

process will be repeated until the request reaches the destination. Pastry find a closer node by matching the most significant bits of the resource with other nodes since Pastry implements the Plaxton, Rajaraman and Richa (PRR) [72] tree scheme to locate items in the overlay.

Each node in Pastry will keep three tables: a routing table, a neighbourhood set and a leaf set as following:

- **Routing Table:** The routing table in Pastry nodes, with ID base of  $2^b$ , holds  $\log_b N$  rows, where  $N$  is the size of the network. Each row holds  $b - 1$  entries where every entry refers to a node which shares the first  $r$  bits of the Node ID but the  $(r + 1)^{\text{th}}$  bit does not. In Pastry,  $b$  is a parameter that control the size of the routing table and can be used as trade off between size and overhead vs. the maximum hop count. So, when we set the value of  $b$ , it will effect both of them as following:

$$\text{Routing Tables Entries} = (\log_b N) \times (2^b - 1) \quad (2.1)$$

$$\text{Maximum Routing Hop} = \log_b N \quad (2.2)$$

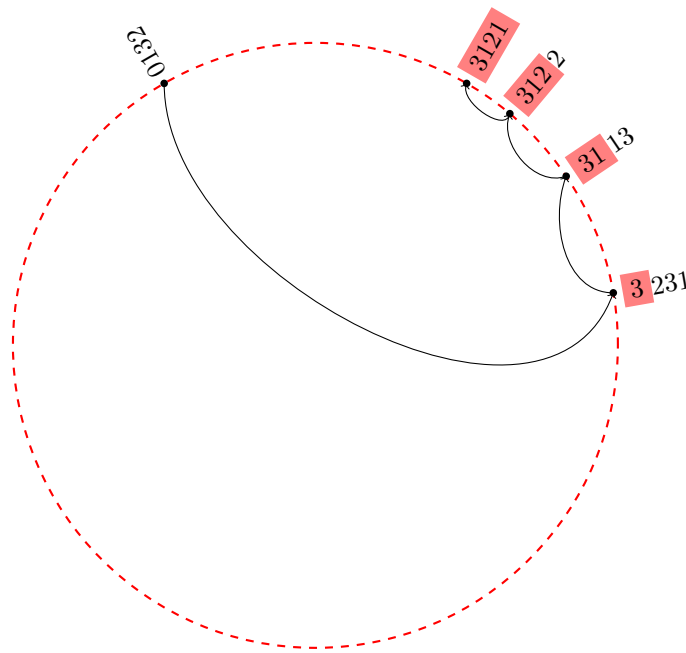


Figure 2.7: Example of routing a query from node 0132 to node 3121 in Pastry

Since any node with the correct prefix can be used as entry in the routing table, Pastry will substitute them with closer nodes. Pastry will update its routing table by requesting the routing table of other peers in the overlay and ping the entries in its routing table. From the ping results, it will try to substitute the entry in its routing table with others that have lower ping i.e. closer. Table 2.3 shows an example of the routing table of a peer in the pastry network with ID of 12030213. It can be seen that at row  $r$ , the list contains a peer that shares the first  $r$  digits which are highlighted in the table. The  $i + 1$  digit will be the number of the column except for the cell which shares  $r + 1$  digit which will be expanded in the next row.

- **Neighbourhood Set:** This set contains a list of the  $m$  closest nodes. Pastry allow application to define a distance function that will be used to calculate the closest nodes.
- **Leaf Set:** This set holds a list of  $k$  nodes whose IDs are closest and centred around the local node ID. Also, an item in the set  $i$  is replicated  $k$  times with  $\frac{k}{2}$  replica preceding  $i$  and  $\frac{k}{2}$  after.

For a request sent to a destination  $d$ , a Pastry node will first check if  $d$  is in the Leaf Set. If so, the request will be forwarded directly. Otherwise, the node will check its Routing Table to find a node which shares more leading bits with  $d$  than the local node. If such a node is found, it will be forwarded the request. If not, the local node will try to find a node in its Routing Table that shares a number of leading bits with  $d$  similar to the local node but numerically closer to  $d$ . This process will be repeated until the request reaches  $d$  in no more than  $O(\log_{2B} N)$ .

For example, to route a query from 0132 to 3121 as seen in Figure 2.7, the query will hop across the overlay. In the example, the query takes the maximum number of hops. With each hop, the query is getting closer to the destination and matching more leading digits. Since this was the worst case scenarios where the routing table of every peer on the query route was not able to match more than one digit, the query can achieve better hop count by matching more than one digit.

For the purpose of the reference implementation of ONM, we have chosen Pastry to be used as the overlay. This is mainly due to its widespread use and the existence of an ALM algorithm, i.e. Scribe [73], that can be used with it.

**CHORD** A consistency hash function, such as Secure Hash Algorithm 1 (SHA1), is used in Chord to consistently assigns nodes and data a unique  $m$ -bit identifier. This is done by taking the output of the hash modulo  $2^m$ . Then, a structure is formed by placing a the node in a ring structure by the order of their identifiers. Lastly, data items are assigned to the node with the same identifier or the first node greater than it.

For a overlay of size  $N$ , each node will maintain a routing table with size of  $O(\log_2 n)$ . The routing table will hold  $k$  successors, or nodes succeeding it in the ring. Also, it will hold a list of nodes chosen at logarithmically increasing distance around the ring. This list is known as the finger list. For a node with identifier  $n$ , The  $i^{\text{th}}$  node in the finger list will be pointing to the first node with an identifier that is equal to or greater than  $n + 2^{i-1}$ . The logarithmic distance between nodes in the finger list will allow a node to have a better view of its

nearest neighbours. For nodes further away, the finger list have a more disperse overview of them.

To route a query to a destination  $d$ , the sender will select the first node from the finger list whose identifier is most immediately precedes  $d$ . It, in turn, will do the same and push the query closer to  $d$ . As a query is forwarded across the overlay and get closer to its destination, the more likely it will find a link to the destination. This process will be repeated until it reach a node that have  $d$  in its successors. As a worst case, it will take a request  $O(\log_2 n)$  hops.

An example for the overlay is shown in in Figure 2.8. In the figure, we can see node  $a$ , which has a finger table of 7 entries. Each entry will have a reference to a node identifier and it location i.e. IP address. At each entry  $i$ , the location of the at that location or the first one following it. For example, in the finger table at the entry of  $i = 5$ , where looking for the node that is at reference 32 or directly after it. In that case it is node  $g$ .

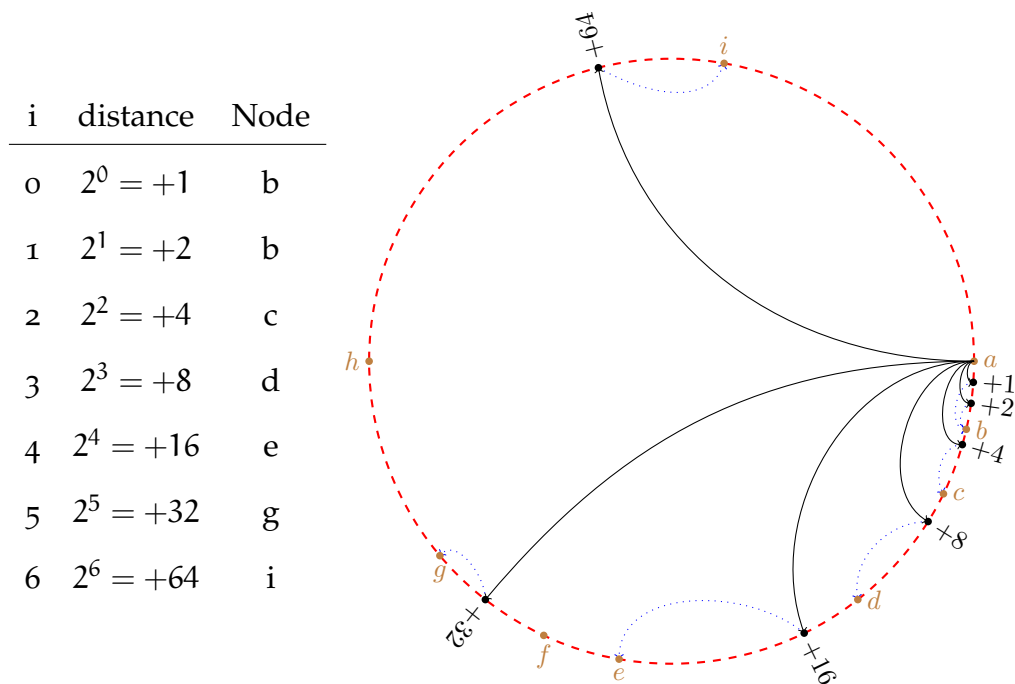


Figure 2.8: Example of the finger table in Chord for node  $a$

**KADEMLIA** Kademlia [71] uses a 160-bits address space to assign identifier for peers and services. Similar to Pastry and Chord, Kademlia uses a hash function to calculate their identifier. With each hop, the peers will move the query one hop closer to the destination. The distance between nodes are calculated using an Exclusive OR (XOR) function.

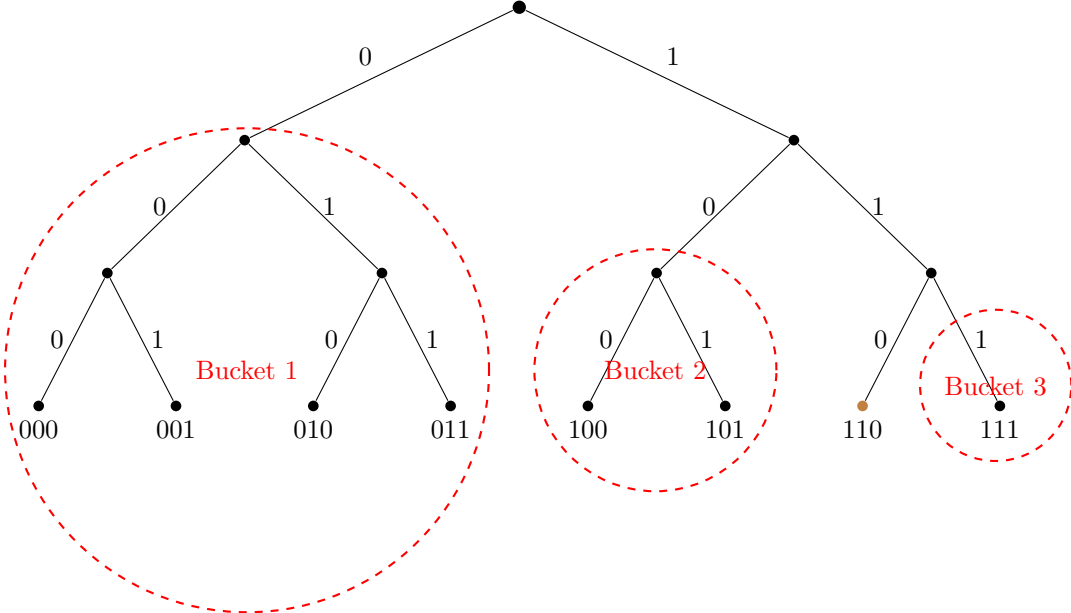


Figure 2.9: Kademlia Routing Buckets for node 110

The routing table of Kademlia consists of  $b$  buckets, where  $b$  is the bit-length of the identifier. In the case of Kademlia,  $b$  is 160. For each distance  $d$ , a bucket exists whose nodes do not share the  $d^{\text{th}}$  bit but the most significant  $d$ -bits are similar. The size of the buckets is referred to as  $k$ . A Kademlia peer will keep adding nodes that it encounters to the list until  $k$  is reached. For a peer, it will have less value to choose from with closer buckets allowing it to have a better mapping for nodes in closer buckets.

Figure 2.9 shows an example of the way that the buckets are assigned in Kademlia. It can be seen that the closer the nodes buckets the smaller the buckets allowing  $k$  to be a larger part of it.

**CAN** Content Addressable Network (CAN) [42] uses a virtual  $d$ -dimensional Cartesian coordinate space. As nodes join the overlay, it will be assigned a zone



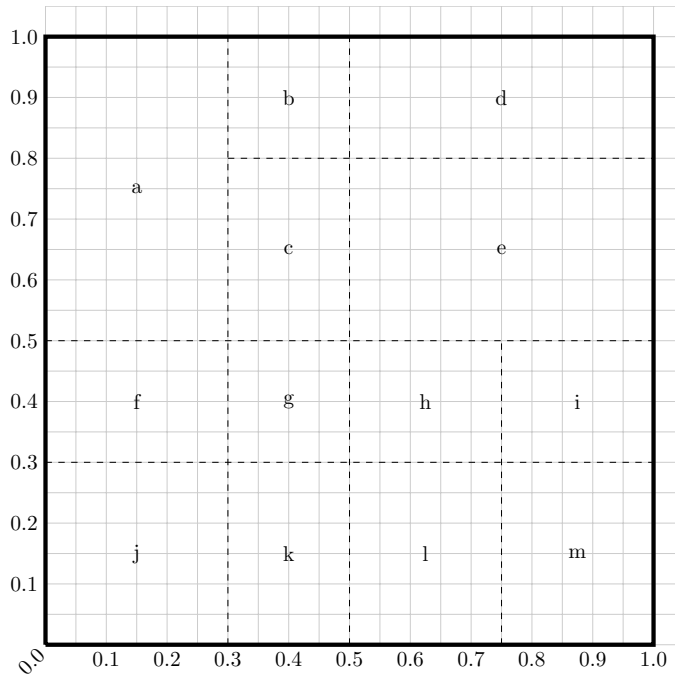


Figure 2.10: The CAN overlay divided into zone in a 2 dimensional coordinate space

of the space. Data and services will be mapped using a hash function to points in the coordinate space and will be assigned to the peer owning the zone. In a CAN overlay with  $d$  dimensions, each peer has at least  $2d$  neighbours: one to move forward across the dimension and another to go backward. Each CAN peer will keep a coordinate routing table that points to its immediate neighbours. Two peers are consider neighbours if their coordinate overlap on a  $d - 1$  dimensions. So, to forward a query to a destination, the hop will be forwarded in a straight line through the coordinate space to the distention. Each peer will send a periodic update regarding it status to its neighbours. When the peer fail, its neighbours will detect the failure and takeover the empty zone in the coordinate space.

An example of a CAN overlay can be seen in Figure 2.10. The figure shows a 2 dimensional coordinate space divided into 13 zones. For example, zone c have 4 neighbours, two for each dimension: a,b,e and g.

### 2.2.3.2 One Hop

Here, the overlay, when converged, is a fully meshed graph of nodes. Since all nodes are aware of each other and can communicate directly, it will take

a single hop to reach any needed resource in the overlay. While this yields a high performance of the lookup, it requires a high number of control overhead messages that needs to propagate to each node in the overlay. This become expensive when the number of nodes increases resulting in a scalability issues. An example of this type is Epichord [74] and OneHop [75].

**D1HT** One example of one hop overlays is D1HT [76]. D1HT tries to achieve a one hop overlay while trying to conserve on the maintenance traffic. It will try to propagate events through out the network. Similar to many structured overlays, D1HT maintain a ring structure. It will assign a  $m$ -bit identifier for node and resources using one-way functions or hashing. Then it will order the node in the ring using the module  $2^m$  of the identifier. The resources items are assigned to the next node in the ring to the resource identification.

To achieve the one hop routing, each node in the overlay have a routing table that links to every other nodes in the overlay. Thus allowing each node to communicate with other nodes directly.

D1HT uses a protocol for to maintain the routing table called Event Detection and Report Algorithm (**EDRA**). This protocol allows for events to be propagated the rest of the overlay. For this process to minimise the bandwidth overhead and balance the load across multiple nodes, the event takes a logarithmic time to reach the whole network.

For a peer to join the overlay, it obtain its identifier using a hash function to allow it to place itself in the ring structure. then, to keep track of current peers in the overlay, the peers use **EDRA** to disseminate the event the rest of the network. According to **EDRA**, a peer sends a regular propagation message at every  $\theta$  seconds. The peer will send  $\rho$  copies of the message where  $\rho = \log_2(N)$ , where  $N$  is the number of peers in the overlay. A Time To Live (**TTL**) value of  $\ell$  is assigned to each message, where  $0 \leq \ell < \rho$ . Also, peers will forward the received events messages as long as the **TTL** is valid.

Figure 2.11 illustrate an example of a network with 9 peers,  $N = 9$  and  $\rho = 4$ . Po advertised an event by sending  $\rho$  copies of the message with varying values

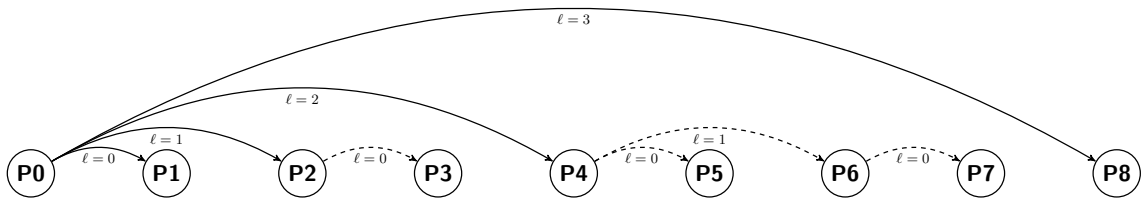


Figure 2.11: An Example of an event propagating using EDRA in D1HT

of  $\ell$  from 0 to  $\rho - 1$ , 0 to 3 in this case. A copy with  $\ell = i$  will be sent to the  $2^i$ th node. So, event messages with  $\ell = 0, 1, 2, 3$  will be sent to the 1st, 2nd, 4th and 8th nodes respectively. If  $\ell > 0$  and the message was received within the last  $\theta$  seconds, the receiving nodes in turn will forward the message after decreasing its  $\ell$ . Note that multiple messages are aggregated and sent as a single message to each receiving end.

**EPICHORD** Epichord [74] is a structured overlay that utilise DHT. Epichord can achieve a one hop performance under an intensive look-up loads. In the worst case, Epichord will need  $O(\log_2 n)$  hops. Epichord in based on Chord (discussed in 2.2.3.1. Epichord can achieve a  $O(1)$  with intensive workload on the network [77, 78, 79]. Epichord achieves this by caching peers information and updating the cache by observing lookup traffic.

### 2.2.3.3 Variable Hop

Here, the structure of the overlay is not symmetric. It will allow nodes to have any value from a one hop to a multi-hop performance. If a node can handle the extra resource requirements, it can increase its routing table. The overlay will take some factors into account when deciding the connectivity of the graph. For example, Accordian [80], focuses on reducing the latency of the lookup. It will do so by searching for new nodes and evicting nodes that are assumed to be dead. The size of the lookup table will depends on the bandwidth of the nodes. Another example of variable hop overlay is Chameleon [81]. Here, nodes assess their bandwidth availability and decide if they are Low Bandwidth L-nodes or

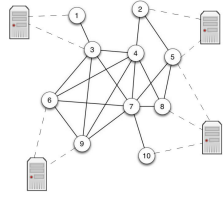
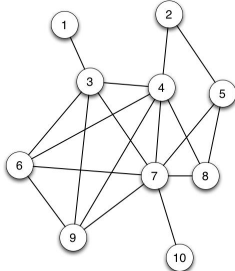
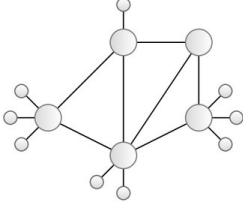
	Unstructured		
	Centralized	Pure	Hybrid
Resolve Lookup:	Central Servers	Peers	Super Nodes
Examples:	Napster, KaZZa	Gnutella	GIA
Sample Structure:			

Table 2.4: Comparing Different Types of Unstructured Peer-To-Peer Networks

High Bandwidth H-Nodes. Then, a two tier overlay is built, one for each type. The H-nodes will have a one hop lookup while the L-nodes will complete the lookups between one hop and  $O(\log_2 n)$ .

#### 2.2.4 Unstructured

The first generation of P2P systems was employing unstructured algorithm that do not have much assumptions about the connectivity of the nodes or the shape of the resulting graph. In this model of P2P overlay, the connection between nodes are established randomly. Nodes can form and remove links as they see fit without an overall structure imposed. The lack of restrictions allows node to easily join the overlay by copying the routing table of an existing peer.

Table 2.4 shows the three main types of unstructured peer to peer algorithms. Centralised algorithms rely on the existence of a central servers that will be aware of the location of resources in the graph and will resolve the lookups made by the peers.

On the other hand, some algorithm will use a pure decentralised peers. So, for queries to propagate throughout the overlay, mainly two ways are be used:

**RANDOM WALK** A node when receiving a query, will forward it to a random peer or peers other than the source of the query.

**FLOODING** A node when receiving a query will flood it all other nodes connected to it.

More recent unstructured P2P overlays utilise hybrid approaches exist were some peers play more important roles than others. For example, GIA [64] allows nodes which can handle more lookup requests to act as a super peers. In GIA, the lookup will propagate using weighted version of Random Walk approach: Biased Random Walk. Just like Random walk, Biased Random Walk will direct lookup queries toward high capacity nodes which will have a higher probability of answering the request [64].

Since unstructured overlays does not have an overview of the shape of the network, it will have to create multiple look up queries until a resource is found or the query expired. A request is expired to prevent it from traversing the network for too many hops e.g. when stuck in a loop. The node initiating the request specify a value of Time To Live (TTL). At each hop of the overlay, this value will be decreased by 1 before forwarding it. When TTL reaches 0, the request is deleted. This will limit the scalability of the network, as the traffic will increase as nodes join in.

Unless the whole network is traversed, a node cannot have any guarantee about the existence of a resource or how many hops it will take to find it. An example of the unstructured protocols are Freenet [82] and Gnutella 0.4 [83].

#### 2.2.5 *Differences between Structured and Unstructured Overlays*

Further key differences between structured and unstructured algorithms are [84]:

- Structured algorithms are aimed at exact search while unstructured ones are better used for keyword and wildcard searches. In our approach, nodes

will lookup exact matches for Primary Nodes in the network. Wildcard searches are not required.

- Unstructured algorithms work well if the resources are highly replicated. In our approach, we want to minimise replication traffic. This, in turn, will reduce the overhead in the links connecting islands as they have higher costs than LAN traffic.
- Unstructured algorithms work well with highly transient networks. However, we assume minimum peer movements between islands. This is especially true when considering core network devices to operate as Primary Nodes.
- Unstructured overlay algorithms use best-effort lookup mechanisms. In our proposed approach, we opted for the more reliable lookup offered by DHT as employed in structured overlays.

Due to these limitations, we have chosen to discard unstructured peer-to-peer algorithms and focus our research on structured approaches.

### 2.3 AUTOMATIC MULTICAST TUNNELLING (AMT)

AMT allows for connectivity between multicast islands. Also, it allows users, connected to unicast-only network, to join in multicast groups. Using AMT, multicast traffic can be exchanged through a tunnel that is built automatically. Using the tunnel, the traffic will be encapsulated in User Datagram Protocol (UDP) packets which will be sent as a unicast through the unicast only network. The AMT tunnel consists of four parts: AMT site, AMT relay, AMT gateway and AMT Pseudo-Interface. The AMT site is a multicast network or a host that has a gateway served by AMT gateway. AMT relay is a multicast router that route traffic between the AMT site and the native multicast backbone. This relay terminates one end of the AMT tunnel. Also, it will encapsulate multicast traffic in the tunnel. AMT gateway is a host or a gateway that is not connected directly to the

native multicast backbone but has an AMT Pseudo-Interface. The gateway will terminate the other end of the AMT tunnel. Also, it will decapsulate multicast traffic from the tunnel. Finally, AMT Pseudo-Interface is the tunnel end-point. The Pseudo-Interface is needed since the tunnel will connect to unicast-only network. [85]

The AMT works using the client-server approach. Without the use of AMT, if the host connected to the unicast-only network were to try to send IGMP membership update messages, the network would drop these packets due to the fact that the network does not support multicast. Alternatively, a process in these hosts may directly intercept such requests. In order to setup an AMT tunnel, an AMT request will be sent toward the AMT relay. This will establish a tunnel between the gateway and the relay using a 3-way handshake. With the tunnel in place, any IGMP membership update messages will be encapsulated in the AMT tunnel. The AMT relay will decapsulate the IGMP membership report and will trigger PIM join toward the source. After the tunnel is setup, any further IGMP messages will be encapsulated in the tunnel directly. Finally, the AMT relay will send any multicast traffic to the hosts that are interested by encapsulating them in the tunnel.

### 2.3.1 *AMT Operation*

In order for AMT to work, the AMT Relays are assigned an address with a special prefix so that gateways can find them. The Internet Systems Consortium ISC has assigned the prefix 154.17.0.0/16 [86]. This address will be advertised in the unicast-only network using the routing protocol used in the unicast only network e.g. OSPF, EIGRP. When an AMT Gateway wishes to connect to an AMT Relay, the gateway will send an AMT Relay Discovery Message to one of the relays in this prefix. This AMT message is sent to the UDP Port 2268 and includes a nonce.

When an AMT Relay receives the AMT Relay Discovery Message, it will answer the gateway with an AMT Relay Advertisement Message. This message includes the unique IP Address for the Relay i.e. the unicast address for the AMT relay. This

address will be used in any further communication to the Relay. The Message will contain the same nonce that has been sent in the AMT Relay Discovery message.

After the gateway receives the advertisement message, the 3-way handshake can start. This process consists of the exchange of three messages between the relay and the gateway. The first message is an AMT Request message to the Relay that includes a new Nonce.

Then, the AMT Relay responds with an AMT Query that includes the same Nonce from the gateway Request and a Message Authentication Code (MAC). The MAC will be calculated also in every subsequent message. The AMT Query will encapsulate an IGMP Membership Query.

If the AMT gateway wants to join a group, it will respond with an AMT Membership Update. This update will include the original nonce from the AMT Request and it will encapsulate an IGMPv3 Membership update.

After the tunnel has been set up, the flow of traffic will begin. The relay will use the pseudo-interface in the gateway as the destination for the downstream.

When the tunnel is already up and the same gateway wishes to join another group, it will enter the 3-way handshake directly.

An example of the flow of the messages can be seen in figure 2.12

After the establishing of a tunnel, the AMT gateway will start acting as a IGMP server to the hosting connecting to it. So, the AMT Gateway will periodically send queries to refresh the membership status for all the groups in the network. Similarly, The relationship between the AMT gateway and the Relay is according to the IGMPv3 protocol. This means that the Gateway will send a periodic AMT Membership Updates to the Relay. Also, if the gateway wishes to leave a group, it will send an update to leave the group to the relay. If the relay did not receive the messages from the tunnel it will time out.

### 2.3.2 *Advantages of AMT*

For Hybrid Multicast, we will utilise AMT to connect multicast islands. Using AMT will provide us with the following advantages:



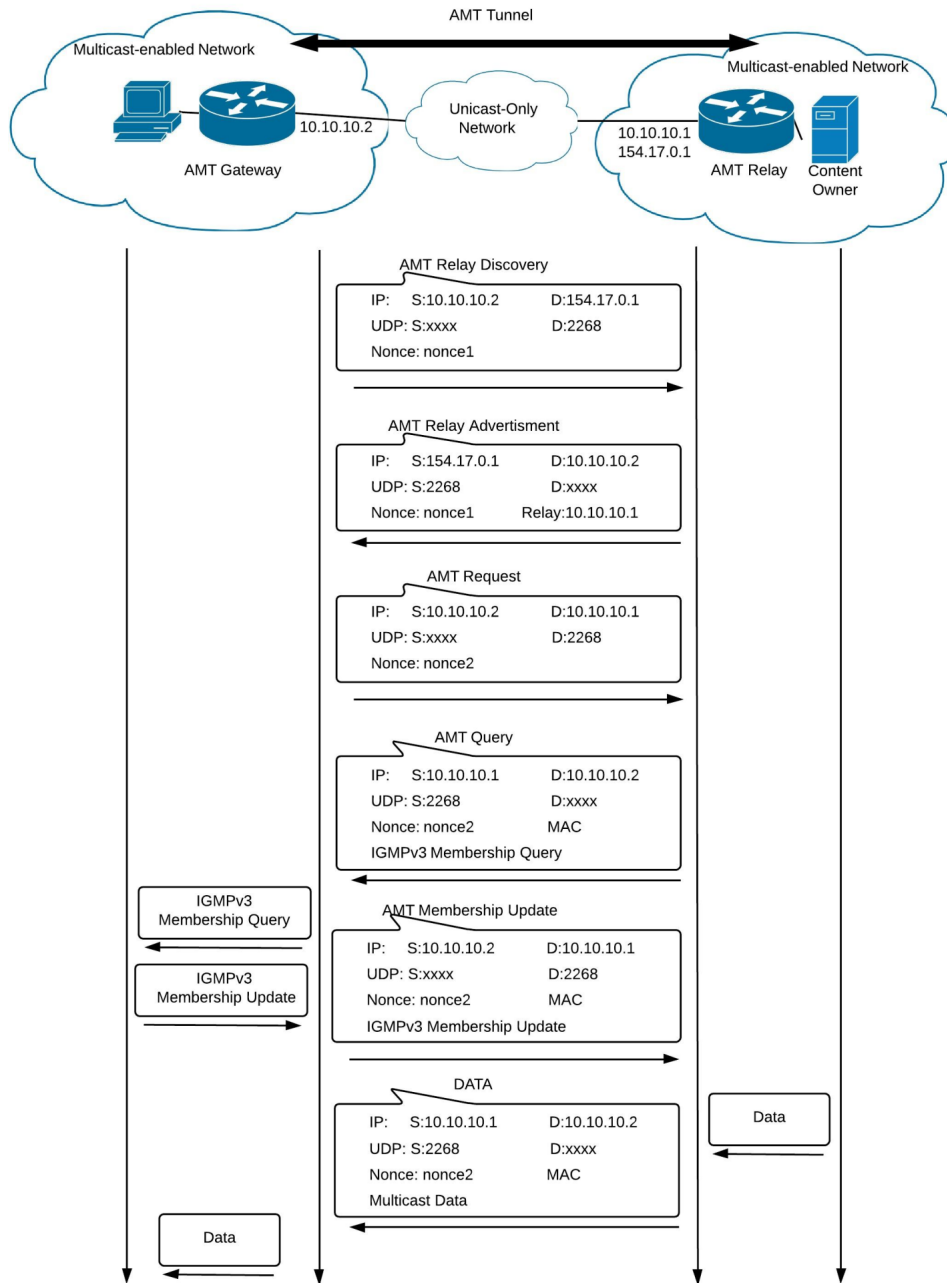


Figure 2.12: The Flow of AMT messages

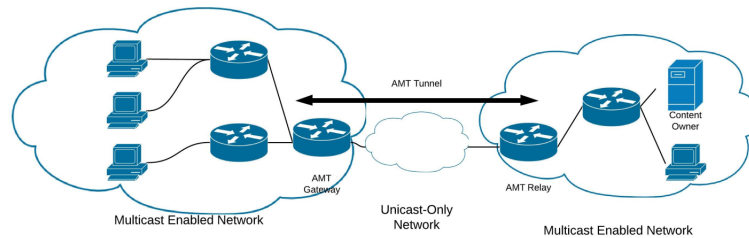


Figure 2.13: Multicast Islands Connected Using AMT

- AMT will keep Layer 4 header intact. This will help when performing traffic shaping and Quality of Service in the network.
- AMT will allow for AMT clients who are not part of the overlay to join the network. This will give our approach interoperability since AMT is standardised in RFC7450.
- AMT supports for authentication that can be utilised to add security and robustness to the system.

Due to all of these mentioned advantages, we will use AMT in our proposed approach to connect Multicast Islands.

## 2.4 SUMMARY

In this chapter, P<sub>2</sub>P and its advantages was discussed. Moreover, different types of P<sub>2</sub>P protocols and how to use them was reviewed. Next, multicast and its protocols were discussed. Also, the chapter discussed different ways to achieve multicasting and compared their advantages and disadvantages.

While multiple P<sub>2</sub>P protocols were introduced, for our purposes, any one of these types of overlays will be sufficient as long as there is an ALM protocol that works with it. However, choosing a Multiple Hop approach may have an advantage of lower management overhead. Some other approaches that can be used are: CAN Multicast [87] and NICE [19]. However, since Pastry was validated for Oversim [88] in [89], which is the base for the module we have developed in [4], we have chosen Pastry and Scribe for our reference implementation.

## OPPORTUNISTIC NATIVE MULTICAST

---

### 3.1 INTRODUCTION

As discussed in Chapter 2, there are many ways to accomplish one-to-many or many-to-many communication. There are two main approaches used: Native Multicast (NM) Application Layer Multicast (ALM). Each one has its own advantages and trade off as discussed in Sections 2.1.2 and 2.1.3. Also, there exists hybrid approaches which tries to combine both approaches, discussed in 2.1.4. In this chapter, we discussed our proposed hybrid multicast approach which is termed Opportunistic Native Multicast (ONM). The alternative to using opportunistic approach is to manually declare the capabilities of each island by a network administrators. However, this approach requires beforehand knowledge of the network and to update it manually when changes happen. Also, ONM utilise the use of AMT tunnels which are more suitable for carrying Multicast traffic between islands as discussed in 2.3.2. ONM takes advantage of cross layer awareness combining information from different layers to make decisions. Network topology and physical connectivity are only available in the network layer of the OSI stack. Lower level information is needed for the discovery of islands and the mechanism to elect nodes, which will be discussed in detail in section 3.3. By accessing the information available in the lower layers (Network and Data Link), ALM peers are aware of the status of the underlay including the support of native multicast and the existence of other peers in the same multicast domain. Peers joining the same ALM group who are also located in the same native multicast island communicate more efficiently using native multicast for intra-island communication. Figure 3.1 shows the topology of connections between nodes at

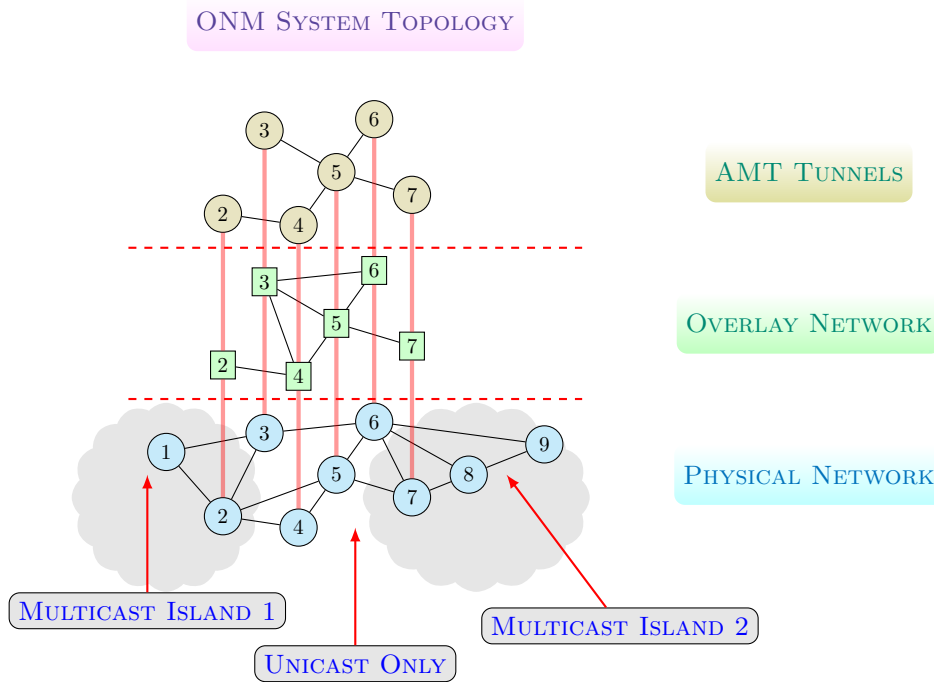
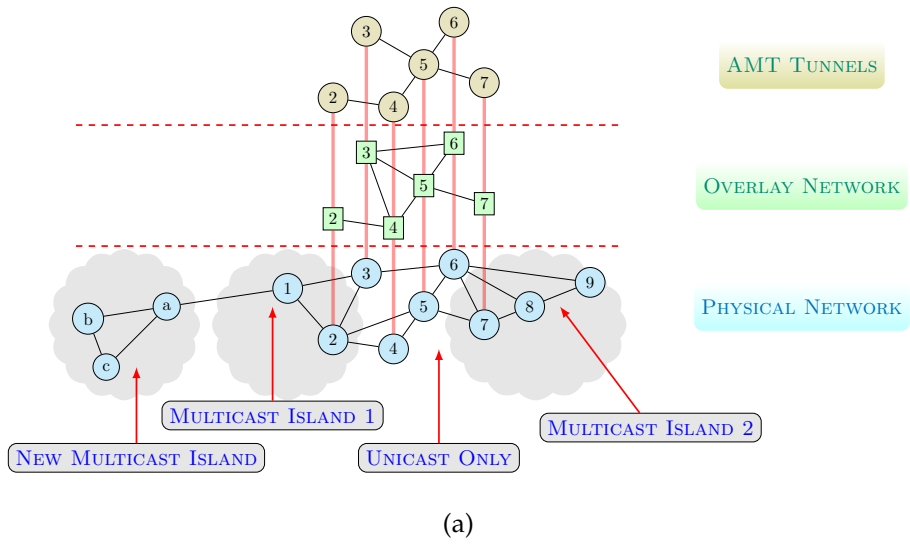


Figure 3.1: ONM System Topology

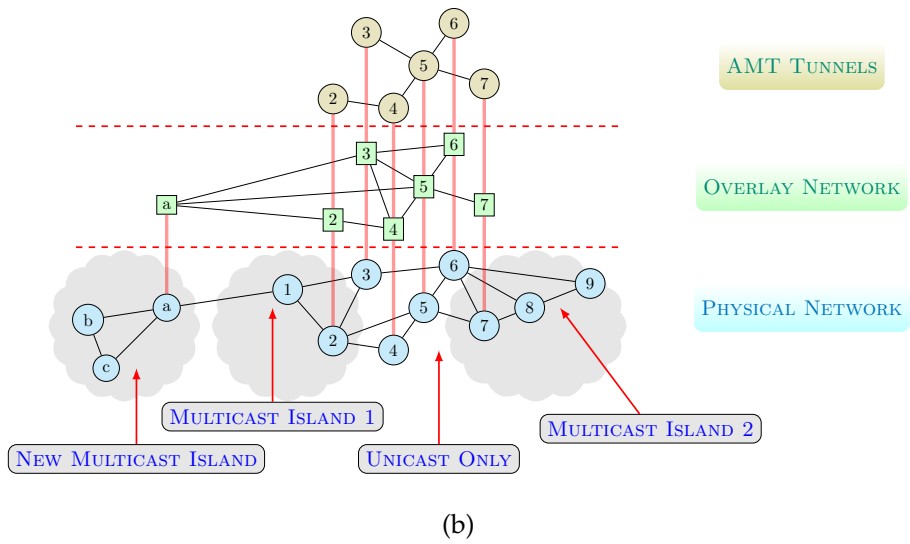
different layers. Note how the connection between Node 1 and Node 2 is done only using the Native network without using the Overlay.

### 3.1.1 Joining ONM

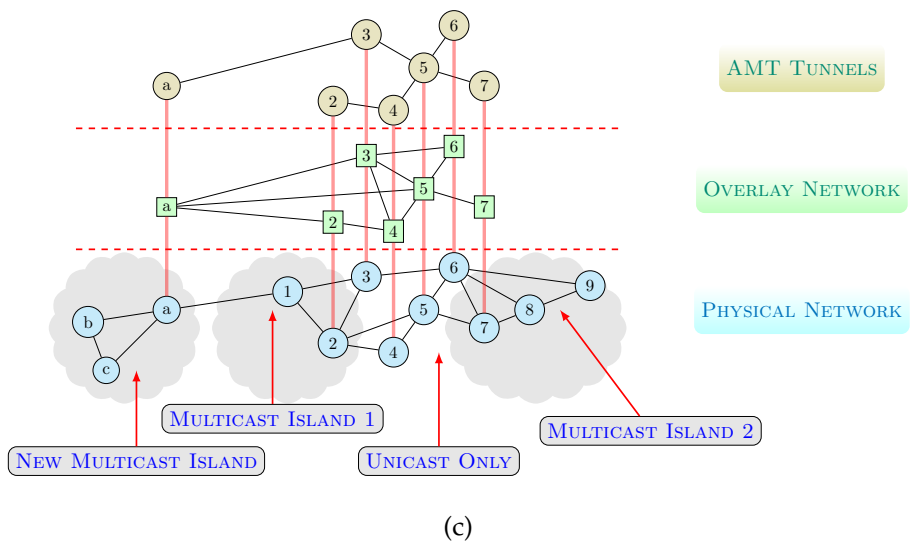
In this section, we will review how can an island join Opportunistic Native Multicast (ONM). Figure 3.2 shows the different stages of this process. First, as can be seen in the subfigure a), the island is connected to rest of the network with a unicast only link. This will prevent Native Multicast traffic from crossing this link. This will result in the island being isolated and could not send or receive Native Multicast traffic with the rest of islands. At this stage, the island nodes will elect a Primary Node (PN), in this case it is Node A. More information on how the election process is carried out will be at Section 3.3. Then, the Primary Node (PN) will move to the next stage shown in subfigure b). At this stage, the PN will join the overlay and connect to the Application Layer Multicast (ALM). Here, the PN will be able to receive and send multicast messages to the rest of



(a)



(b)



(c)

Figure 3.2: An Island Joining ONM

ONM network using ALM. After that, the new PN will build an AMT tunnel to another PN as shown in subfigure c). Finally, the new island will be part of the ONM multicast tree. Any new multicast message originating the overlay will be relayed to the island by the PN. Also, any multicast message originating inside the island will be relayed to the rest of islands using the AMT tunnels. At this stage, the PN can select one or more nodes to serve as a Secondary Node (SN). More on this process in Section 3.4.

### 3.1.2 ONM Alternatives

One of the earliest attempts to connecting multiple native multicast islands was MBone [9]. MBone was an experimental backbone to carry multicast packets across the internet that requires a specialised hardware and software. MBone related approaches usually required an administrator to set up and maintain tunnels. Moreover, MBone suffered from some issues of access control, security, address allocation and network management[6]. Automatic Multicast Tunnelling (AMT) was proposed to automate the management of tunnels [44]. AMT defines specific devices, Relays and Gateways, in the network to interact with the native multicast protocol and to establish tunnels when needed. As such AMT focuses on connecting island pairs rather than a global unified network. AMT will be discussed in more detail in Section 2.3.

Except for AMT, one common limitation of the listed approaches is that Layer 4 information are encapsulated and hidden. Layer 4 information are needed to do QoS and traffic shaping.

Tunnelling multicast over unicast-only network is an important aspect of Hybrid Multicast. These tunnels play a vital rule in the performance of multicasting. We can classify these techniques by the way these tunnels are built into two types:

**EXPLICIT TUNNELLING** With Explicit Tunnelling, isolated multicast islands are interconnected using manual tunnels such as GRE. This can be done by the network administrators manually where each tunnel is designed and

maintained. In the case of Multicast over GRE, each multicast packet get encapsulated inside a GRE header[46]. This encapsulation results in that the layer 4 information is hidden to the router connecting the two end point of the tunnel. This information is needed for doing some QoS and traffic shaping.

**WITHOUT EXPLICIT TUNNELLING** Here, tunnels will be created and maintained automatically. The shape and topology of these tunnels will change dynamically as the underlying network changes. This could be done using different techniques such as peer-to-peer overlays. We can categorise these protocols into ALM-First (e.g. HM [43], NICE [19]) and Native Multicast-First (e.g. HIPM [47], ASRM [48], UM [49]).

### 3.1.3 *ONM Advantages*

Using Opportunistic Native Multicast (ONM) has many advantages over the use of other approached. In this section, Island Multicast (IM) is the main focus of the comparison. Similar to Opportunistic Native Multicast (ONM), IM uses unicast to connects islands while utilise Native Multicast (NM) to deliver data the island. As discussed in Section 2.1.4.2, IM can be implemented either as Centralised Island Multicast (CIM) or Distributed Island Multicast (DIM). Since CIM requires setting a central nodes manually, DIM is the one that is comparable to Opportunistic Native Multicast (ONM). In this section, these advantages will be listed and discussed. However, this section will leave the performance analyses between these approaches as they will be discussed in 4.10.

- **Layer 4 Header:** Since ONM uses AMT to tunnel traffic between islands, the layer 4 header will be kept intact. This will allow routers to do QoS and traffic shaping to the traffic which is very critical for real time application such as VoIP.

- **Overhead:** IM requires that every pair in the overlay to form a bridge. This increase the requirement on nodes as maintenance traffic is needed for these bridges.
- **Non-Island Nodes:** ONM allows unicast only nodes to join the multicast tree with minimal overhead. In IM, for a unicast only node to join, it will need to form a bridge with every other nodes in the overlay.
- **Standardisation:** ONM uses and supports AMT nodes, which was standardised in IETF RFC7450. This will allow for non-ONM nodes to join existing islands using AMT.
- **Resilience:** ONM can support multiple backup nodes, Secondary Node (SN). These nodes help in improving the recovery speed in the case of Primary Node (PN) failure. Also, they will localise the control traffic as Primary Node (PN) exchanges heartbeat messages with the Secondary Node (SN).
- **Nodes Priorities:** In ONM, multiple factors can be considered when choosing the Primary Node (PN). This allows for more control and better design as can be seen when using age-based factors. However, IM picks nodes by allowing the first node to be the island leader. This can cause issues with large islands as delay can affect the order of messages and contention arises. Opportunistic Native Multicast (ONM) uses priority field to allow nodes to express their suitability and a 64-bit random number to be used in the case of multiple nodes with same priority. In the highly unlikely case of the collision of both fields, the node with lowest IP address will win.

### 3.2 ONM OPERATION

ONM reduces overhead and increases efficiency by allowing for only one peer per island to participate in the overlay network. By having only one node participating in the ALM tree, the number of copies sent in the ALM tree and the control overhead needed to maintain a large number of peers is decreased. This



participating node is called the Primary Node. The Primary Node is responsible for connecting the overlay multicast tree to its island's native multicast tree.

While a single Primary Node is sufficient for the operation of the protocol, it requires that the network keeps track of the Primary Node. However, the Primary Node in each island, as any other networked device, is susceptible to be churned out of the network. In order for the network to discover the disappearance of the primary node, the primary node will be required to multicast hello messages periodically to every node in the network to inform them that the Primary is still active. When the island misses a certain number of 'Hello' messages, it will assume that the node is dead. When the network detects a Primary Node failure, the other nodes in the island will participate in an election process that is costly in time and bandwidth. So, the concept of a Secondary Node is introduced. The Secondary Node is a node selected by the Primary Node to act as its backup.

Introducing Secondary Nodes brings two main advantages:

- The Primary Node does not have to send Hello messages to the whole network as frequently. Instead, the Primary Node will exchange Heartbeat Messages with the Secondary Node instead. Thus, saving bandwidth.
- When the Secondary Node detects that the Primary Node has failed, it will assign itself as a new Primary Node. Subsequently, it will select another Secondary. This will save entering the election process which is costly in terms of traffic volume and time.

For the current implementation of ONM, we use the protocol stack illustrated in Figure 3.3. In the figure, we can see how the used protocols interact with each other:

- **AMT:** The Primary Nodes will encapsulate data between the islands using AMT tunnels. Each Primary Node will act as an AMT Gateway and AMT Relay for the purpose of carrying locally originating data to other islands and relaying data to the island.
- **Scribe:** This protocol is used for creating and maintaining of the ALM tree. The ALM Tree will be built on top of the P2P network.

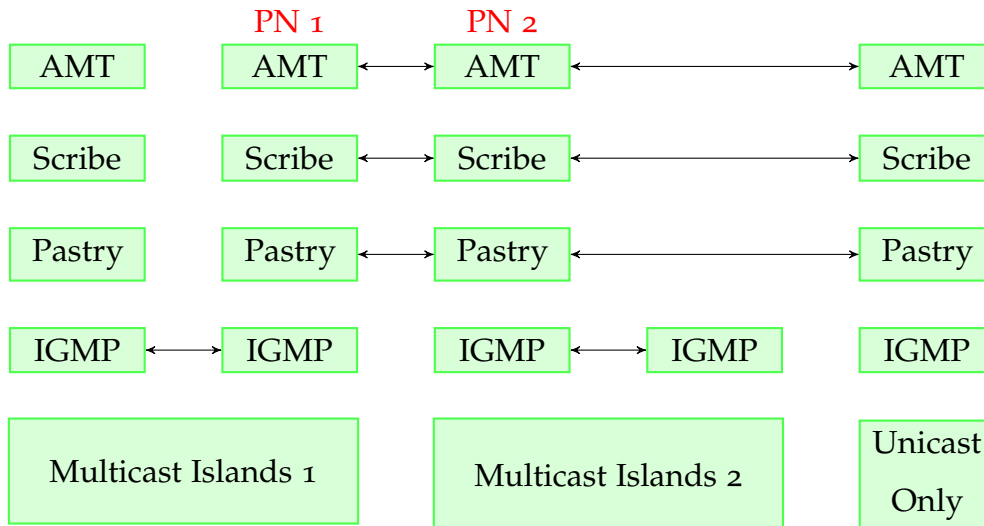


Figure 3.3: The stack of protocol used in ONM

- **Pastry:** Pastry is a structured P2P protocol. It will allow for nodes to join and discovery of other nodes.
- **IGMP:** This protocol is used for multicasting messages inside the island.

So, nodes inside the island will communicate with each other using the IGMP protocol. Also, the primary nodes will join the same ALM tree using Scribe. When a Primary node receives a message originating from the island, it will act as an AMT Gateway and forward the message using AMT to other islands. Alternatively, when the Primary node receives data from the ALM tree, it will act as an AMT Relay and copy the message to the local island.

In Figure 3.3, we can see the different types of nodes that can participate in the operation of ONM. In Multicast Island 1, we can see two nodes. The Primary Node of the island, marked with PN<sub>1</sub>, will communicate with the rest of island using IGMP and Native Multicast. Also, PN<sub>1</sub> will communicate with other Primary Nodes, such as PN<sub>2</sub>, using AMT, Scribe and Pastry. It can be noticed that the other node in the island supports AMT, Scribe and Pastry. This will make the node capable of joining the election and perhaps can be a Secondary Node or future Primary Node.

In Multicast Island 2, PN<sub>2</sub> is acting as the Primary Node for the island. The other node in the island support only Native Multicast. It will not be aware of

the existence of the ONM and will communicate using Native Multicast only. This node can be a node that require Native Multicast to work but is not able or willing to play a role in the ONM.

In the unicast-only network, a node could use AMT, Scribe and Pastry to join the multicast tree and communicate with the rest of the network without the need to Native Multicast support.

### 3.3 PRIMARY ELECTION

Each multicast island elects a primary node that all the nodes in the islands agree on. One option is for the Primary Node to be chosen at random after all of the interested peer nodes join in an election. Alternatively, different criteria could be defined such as the oldest node will win.

Since elections are costly in terms of bandwidth and time, they should be avoided if possible. In ONM, Primary Nodes announce their presence by sending Hello Messages to the native multicast group once every  $T_{\text{hello}}$ . So, when a node joins ONM, it will join the native multicast group and will listen for a specific length of time  $T_{\text{discover}}$ . If after  $T_{\text{discover}}$  it did not detect any Hello messages from the Primary Node, it will assume that none exists. In that case, the node initiates the election process by multicasting an Elect Message to the native multicast group for the duration of  $T_{\text{elect}}$ . If at that time another peer joins the election, the node with the highest priority would be elected. This priority can be determined by age or at random as we discussed before.

Figure 3.4 shows the messages exchanged during the Primary Node election. In this scenario, Node a joins attempts to detect if there is already an Primary Node (PN) in the island. So, Node a will wait for a period of time  $T_{\text{discover}}$ . Then, if there is no active PN, node a will start the election process by sending an  $\text{ONM}_{\text{elect}}$  message to the island. when receiving this message, other nodes will join in the election if:

- They are interested in being the PN.

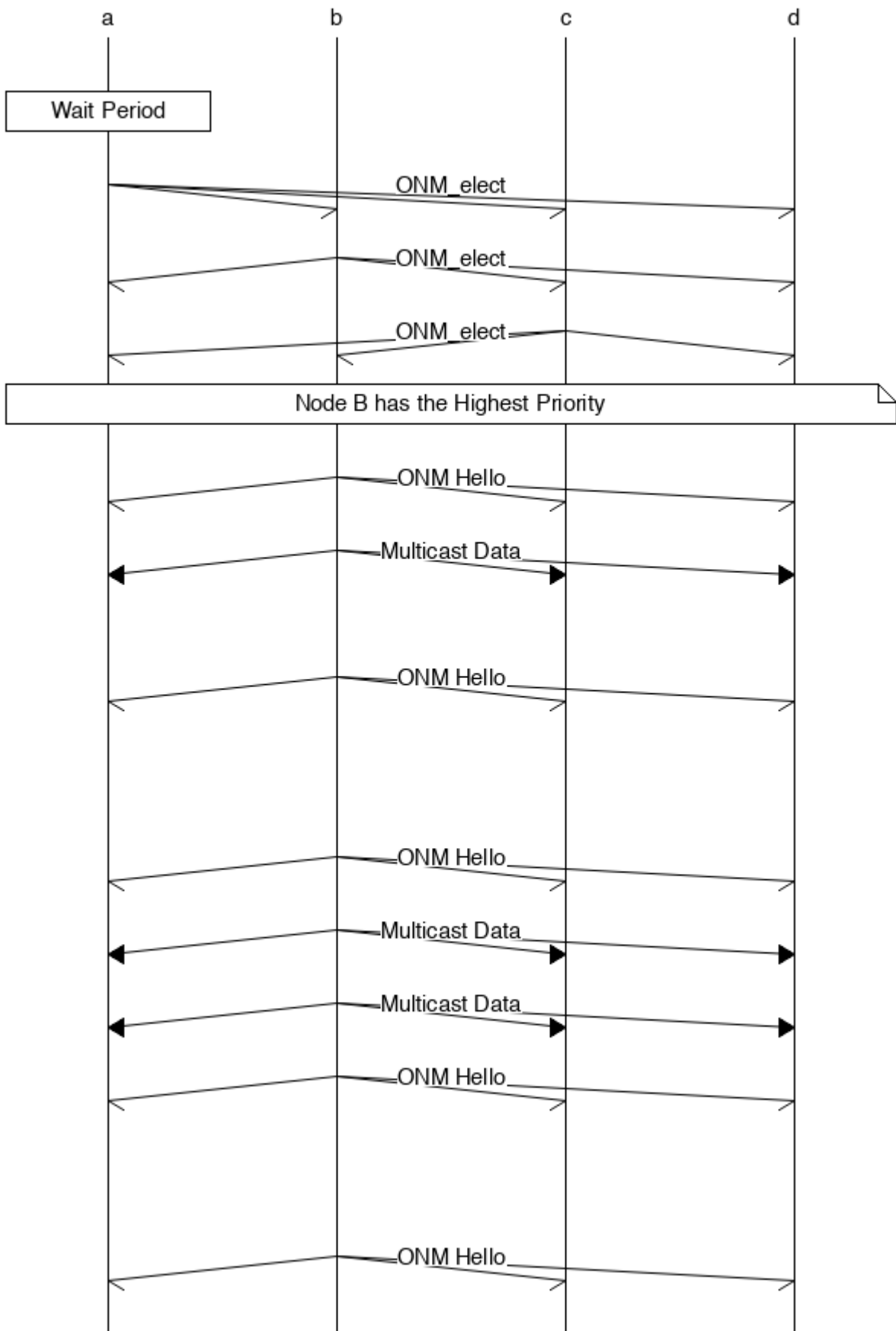


Figure 3.4: Primary Node Election Message Sequence

- They have higher priority .

In this example, Node b and c have expressed their interest by sending the  $ONM_{elect}$  as well. After a certain amount of time has passed since the a node sent its  $ONM_{elect}$  and did not receive any contestants by a higher priority node, it will assume the role of a Primary Node (PN). As PN, it will send a periodic  $ONM_{Hello}$  message every  $T_{hello}$ .

The values of  $T_{discover}$ ,  $T_{elect}$  and  $T_{hello}$  should be selected and tuned to maximise performance and avoid premature election decisions. To avoid nodes failing to detect the presence of a Primary Node,  $T_{discover}$  should be:

$$T_{discover} > T_{hello} \quad (3.1)$$

However, this only applies for perfect network conditions. There is a risk of having a delay or packet loss in the sending of Hello Messages. So, for our implementation we have chosen  $T_{discover}$  to be:

$$T_{discover} = 3 \times T_{hello} \quad (3.2)$$

As for  $T_{elect}$ , it should be chosen so that it allows for enough time for a packet to propagate through the network. In theory, the minimum value that  $T_{elect}$  can have is:

$$T_{elect} < 2 \times T_{trans} + T_{process} \quad (3.3)$$

After a node gets elected as a Primary Node, it will join the overlay network and start to relay the messages coming from the overlay tree into the native multicast tree. Also, any multicast messages originated from the island will be relayed into the overlay tree so it can reach the other islands.

While the Primary Node node is active, it must send periodic Hello Messages to the network. The other nodes in the network should keep track of these messages and detect when the Primary Node has disconnected from the network as quickly as possible. Since network conditions are not always perfect, Hello Messages might fail to reach every node in the network. So, we must keep a balance between fast reaction time and initiation an unneeded elections.

In our implementation, we have decided that a node will assume that the Primary Node is dead when it misses three consecutive Hello Messages.

### 3.4 SECONDARY SELECTION

The Primary Node, can multicast a messages to all nodes in the island to get a list of interested nodes. To speed this process up, The Primary Node will cache the nodes that participated in the election of the Primary Nodes as interested nodes. The choice of the secondary node is fully decided by the Primary Node. The Primary Node can send a request to any node to act as a Secondary Node. If the requested node wants to participate, it will accept the request and initiate the exchange of heartbeats between the Primary and Secondary nodes.

The Primary Node should choose the Secondary Node that exhibit low delay and a good stable connection to avoid triggering unnecessary re-elections.

In this thesis, we propose the use of an age factor that is taking into consideration when the Primary Node selects the Secondary Node. A node's age has been used to improve the choice of neighbours in an Unstructured Peer To Peer networks as in [90]. This assumes that the longer that a node has been on-line, the more likely that it will be available in the future. Also, it will avoid unstable nodes that do not have a stable connection to the network such as mobile nodes. The concept will be discussed in more in details later in sections 4.7.2 and 4.7.5.

In Figure 3.5, the message sequence for selecting a Secondary Node (SN) is shown. First the Primary Node (PN) will send a SN Draft message. This message is multicast to the group receiving the multicast traffic. If a node is interested in acting as SN, it will acknowledge the draft by replaying with Draft Ack message. Then, the PN will pick a node or more to act as a SN.

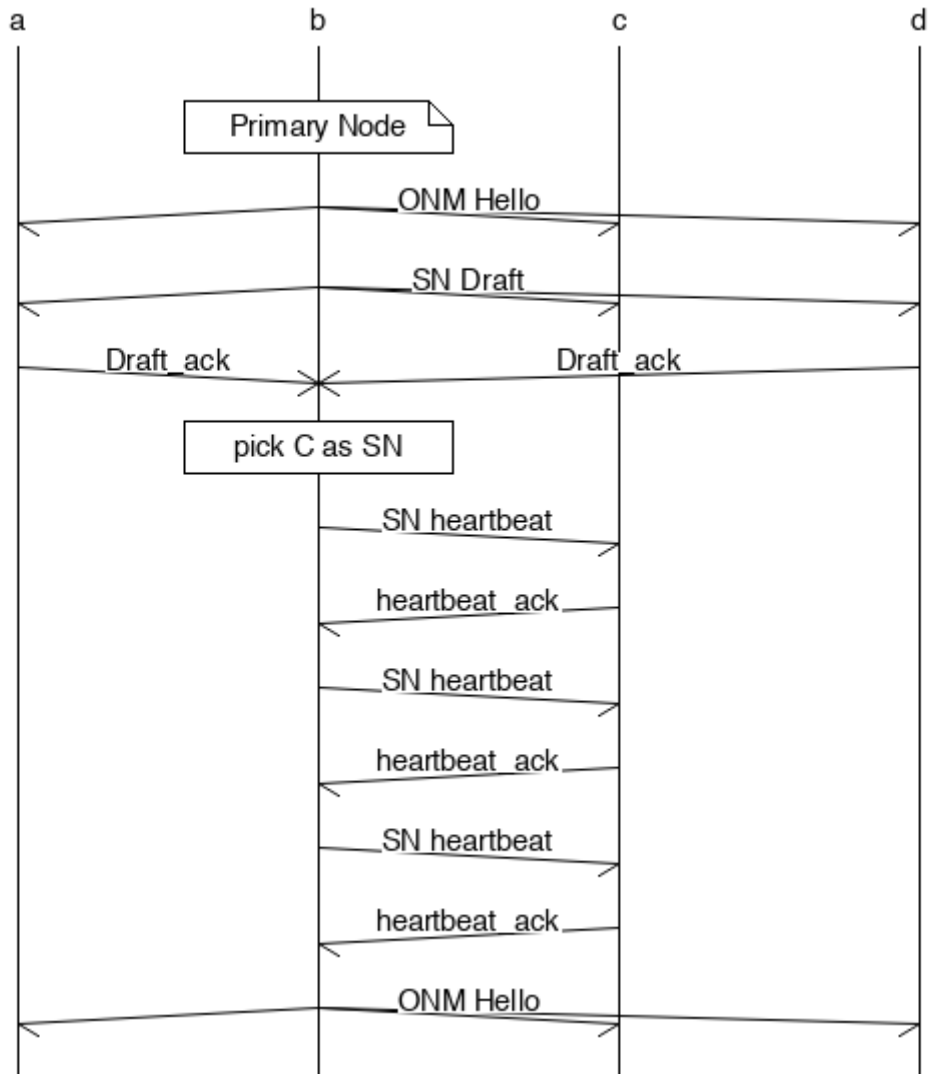


Figure 3.5: Message Sequence for Selecting a Secondary Node (SN)

### 3.5 FAILURE RECOVERY

The Primary Node is central for delivering the multicast traffic. Any failure or misbehaving by this node can cut the entire island from the multicast tree. So, there needs to be a very reliable procedure with low overhead that is able to achieve the following:

- Elect a Primary Node in the case that the network does not have one.
- Allow the Primary Node to select a Secondary Node.
- Keep track of the status of the Primary and the Secondary nodes.
- Allow for the Primary Node to gracefully handover its role to the Secondary Node.
- Detect any failure in the Primary or the Secondary nodes.
- When a Secondary Node fails, the Primary node selects another Secondary node.
- When a Primary node fails, the Secondary Node must become the Primary node and select a new Secondary node.
- When both nodes -the Primary and the Secondary- fail, the network elects another primary node who in turn selects a secondary node.

ONM ensures that the network is behaving properly using different timers. When these timers expire, different events are triggered. Table 3.1 shows the different timers used in ONM and the events that reset them and the events triggered by their expiration.

In Figure 3.6, The message sequence are shown for the case when a PN is churned out. The Secondary Node (SN) heartbeat timeout timers will expire after missing a heartbeat from the PN. After some number of missed heartbeats, the PN will be declared dead. Then, the Secondary Node (SN) will takeover as the new Primary Node (PN).



Table 3.1: The ONM Timers

Timer	Where	Reset Event	Expiration Event
PN Timeout	SN	Receive Heartbeat Message from PN	Takeover
SN Timeout	PN	Receive Heartbeat Message from SN	Select another SN
Election Timer	All	Receive Hello Message from PN	Join election

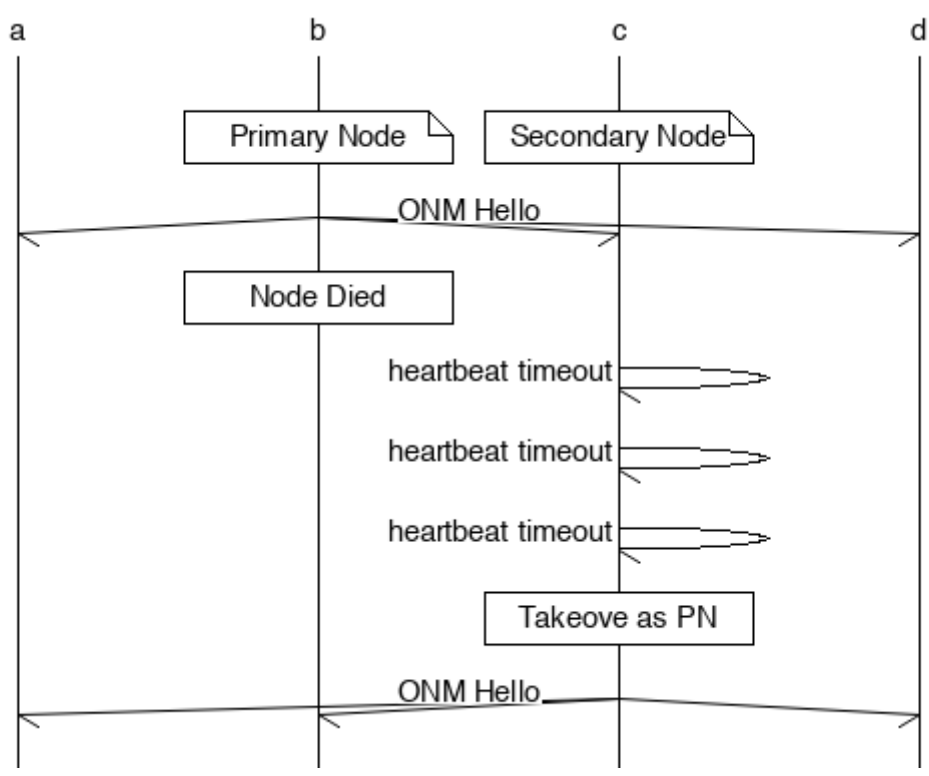


Figure 3.6: The Message Sequence for Recovering after The Churn of The PN

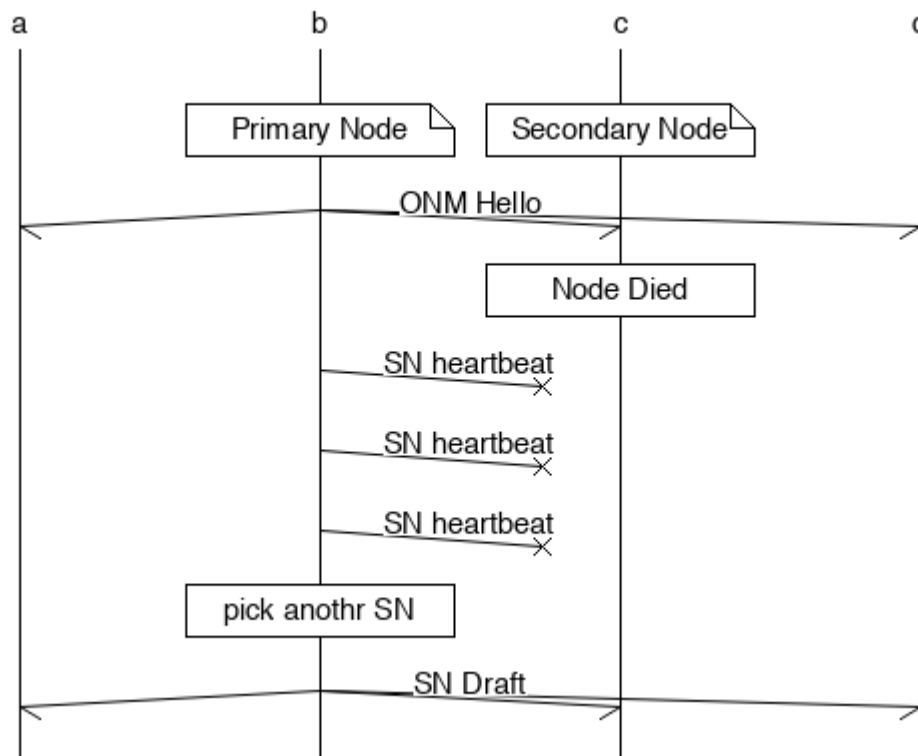


Figure 3.7: The Message Sequence for Recovering after The Churn of The SN

In Figure 3.7, The message sequence are shown for the case when a SN is churned out. The Primary Node (PN) will not receive acknowledgements for its heartbeat messages from the SN. After some number of missed acknowledgements, the SN will be declared dead. Then, the Primary Node (PN) will pick another node to act as a Secondary Node (SN).

The protocol behaves as is illustrated by the finite state machine shown Figure 3.8. There are five states. Every nodes that wishes to participate in the protocol will start in initial state  $S_{\text{initial}}$ .

- $S_{\text{initial}}$ : This is the start state of every node joining the island. At this state, the node will try to join the native multicast group. When the node has joined the native group successfully, the node will move to state  $S_{\text{NM}}$ .
- $S_{\text{NM}}$ : At this state, the node will track the state of the current PN. If no PN is detected, due to failure or being the only active node in the island, the node's state will change to  $S_{\text{Electing}}$ . Alternatively, if it receives a message from the current PN to act as SN it will move to state  $S_{\text{SecondaryNode}}$ .

- $S_{\text{Election}}$ : At the state  $S_{\text{Election}}$ , the node is participating in the election to choose new Primary Node. The nodes participating in the election will exchange the election messages. If the node wins the election it become the Primary Node and will move to  $S_{\text{PrimaryNode}}$  state. Otherwise, it will fallback to to the state  $S_{\text{NM}}$ .
- $S_{\text{PrimaryNode}}$ : At this state, the node is the active Primary Node. When a node enters this state, it will join the ALM tree. The node will relay messages in the ALM tree to the local Native Multicast and vice versa. Also, it will select a node as Secondary Node. When the selected SN fails, The PN will select another one.
- $S_{\text{SecondaryNode}}$ : When the node enters this state, it will join the ALM. At this state, the node is acting as Secondary Node. The node will keep track of the current PN. When a failure is detected, it will become the PN.

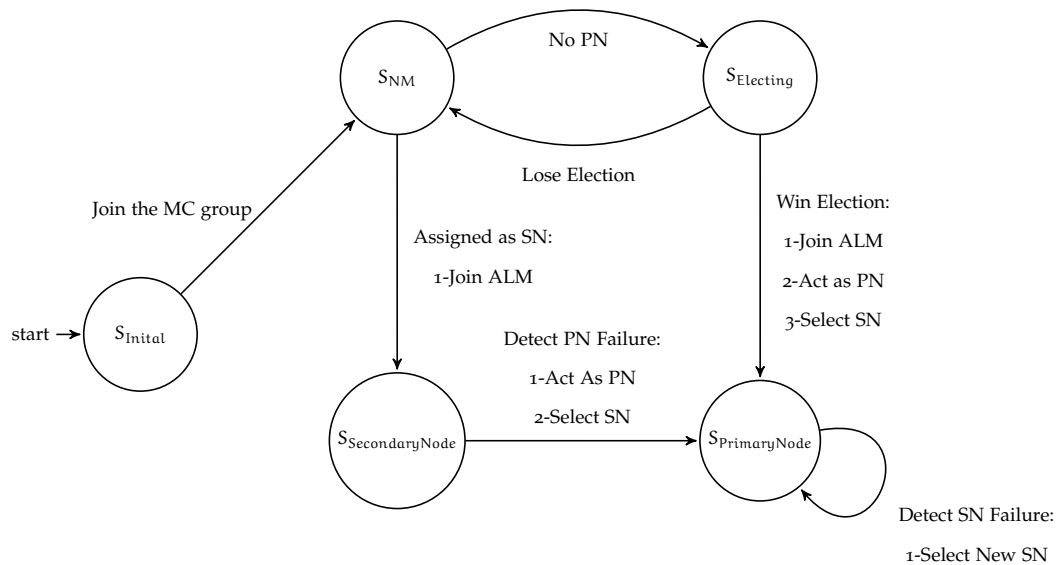


Figure 3.8: ONM Operation FSM

### 3.6 SUMMARY

In this chapter, the operation of the proposed protocol was reviewed. This chapter focused on four main parts of Opportunistic Native Multicast (ONM): The AMT tunnel, the Primary Node (PN) election, the Secondary Node (SN) selection and the failure recovery. Also, we compared Opportunistic Native Multicast (ONM) with the one of main alternative approaches for Hybrid Multicast (HM), which uses the network layer and the application layer in the network stack as discussed in Section 2.1.4, Island Multicast (IM).

## PERFORMANCE EVALUATION OF ONM

---

### 4.1 INTRODUCTION

In our effort to investigate and verify our proposed framework, testing is needed. This can be done using different approaches such as test-beds, simulation or real-life implementation. As scalability and flexibility are one of our main concerns, a simulation approach was chosen.

### 4.2 EXPERIMENTAL METHODOLOGY

#### 4.2.1 *Research Questions*

The questions that the experimentation will try to answer are based on the Research Objectives in Chapter 1. In particular, this section focuses on:

- Demonstrate and verify the operation of the proposed approach.
- Test the efficiency and reliability of delivery. Furthermore, test how the approach will cope with different levels of node churn in the network.
- Verify the the election process of the Primary Node (PN) as discussed in Chapter 3.
- Monitor how the islands can detect when the Primary Node (PN) is not available any more. Also, how can the island react to this event and select another Primary Node (PN).

- Measure the efficiency and recovery speed from Primary Node (PN) failure introduced by allowing the Primary Node (PN) to select a Secondary Node (SN).
- Investigate the frequency of the exchanged heartbeat messages and find the optimum frequency of communication between nodes, between nodes and the Primary Node (PN), and between the Primary Node (PN) and the Secondary Node (SN).
- Test different methods introduced to allow islands to detect the stability of the network and change the frequency of the heartbeats accordingly.

#### 4.2.2 *Benchmark Selection*

Peer-to-Peer networks can be studied using different approaches. In [91], these approaches can be summarised in:

**CRAWLER** Crawlers are specially designed peers that participate in the network.

These nodes are designed to collect data and statistics about the whole network. Multiple crawlers are used to provide more coverage and speed. However, the accuracy of this approach is compromised due to the fact that the crawlers only know about the regions that have been crawled.

**EMULATION** Using emulation can be more effective and comprehensive than crawlers. By deploying a small peer-to-peer network in a controlled environment, emulation can allow us to study the network as a whole. However, the scalability limitations of such approach would not allow for any serious analysing for a large network.

**SIMULATION** By simplifying some assumptions about the network, simulators are able to give us a pretty clear and reproducible study of the peer-to-peer network.

Crawler cannot be used to test its performance since ONM is not implemented in the real world yet. Moreover, emulation can provide more accurate results. However, simulation was chosen for the following reasons:

- Simulation is cheaper and easier to implement than emulation.
- It allows for more room for testing and tweaking and change of configurations to experiment with different settings.
- The Scalable Adaptive Multicast Research Group (SAM RG) within the Internet Research Task Force (IRTF) has identified Oversim as one of the ways for evaluating hybrid multicast. [92].

Also, simulation is used for testing other similar algorithms in the literature. For example, Island Multicast (IM) [50] and Universal Multicast (UM) [49]. So, in our research, we have chosen the simulation approach.

#### 4.2.3 *OMNet++ and INET Framework*

After reviewing different options for simulating computer networks, it was decided that the best choice for me was Omnet++ and oversim for the following reasons:

**EXTENSIBLE:** This is a must feature since there is no implementation for AMT in any existing simulator as of this date.

**MODULAR:** This would help on code reusing some of the existing code for other models and for giving back to open source community one a working code has been reached.

**C++ BASED:** It will allow for more granular control.

**INET MODEL:** This model provides wide range of model of several Internet protocols which

**OVERSIM:** Which provides a way to simulate peer-to-peer overlays.

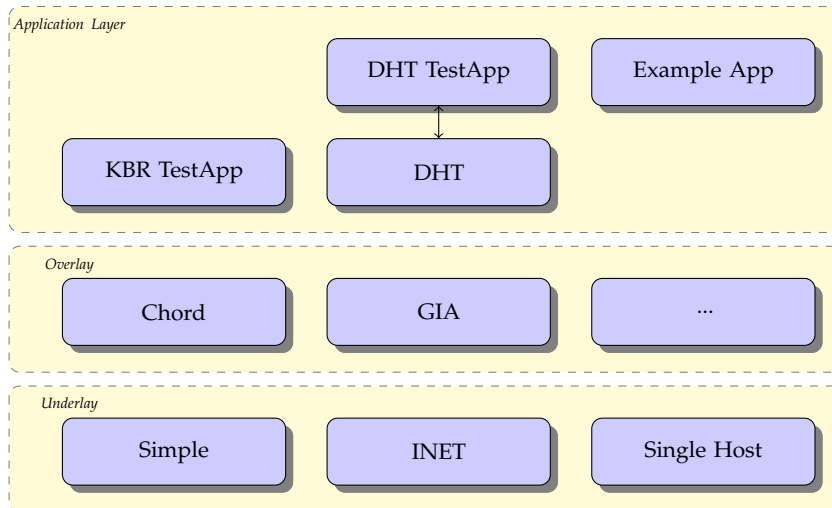


Figure 4.1: OverSim Structure

However, it did not support the needed functionality for my scenarios. While Oversim supports P2P protocols, it lacks the ability to support Native Multicast protocols. Also, it did not have support for AMT tunnels. These two parts are very essential in the operation of ONM. So, I had to modify Oversim simulator to support Native Multicast and AMT tunnels.

The changes done to the simulator has been studied and some results of the initial experimentation with multicasting and AMT testing was done. The results where published in [4].

#### 4.2.4 OverSim

OverSim is an open source peer-to-peer overlay simulator. It is based on the OMNeT++ simulator. Oversim contains several models for structured (e.g. Chord, Kademlia, Pastry) and unstructured (e.g. GIA) peer-to-peer protocols. Because of its modular design, it is very easy to experiment with different protocol easily. As been described in [88], Figure 4.1 shows the architecture of OverSim. We can notice also in Figure 4.1 that the application layer has been divided into two layers or tiers. That will increase the modularity of the simulator.



Parameter	Default Value
Network Size	1000 Nodes
Average Node Lifetime	3,000 seconds
Churn Model	Weibull Distribution
Generated Traffic Sending Interval	5 seconds
Generated Traffic Message Size	1400 byte
Network Topology	Star
Measurement Time	7200 seconds
Repetitions	5
Pastry's Leaf set size	32 nodes
Pastry's Neighbourhood set size	16 nodes
Pastry's Bits per digit	4 bits
Pastry's lookup redundant nodes	4 nodes

Table 4.1: List of default parameters for the simulations

#### 4.2.5 Simulation Setup

Unless stated otherwise, these are the parameters that are used:

- **Network Size:** The default number of participating nodes in the multicast are 1000 nodes.
- **Average Node Lifetime:** Most of the following simulations use multiple values of Average Node Lifetime which define the average time of nodes in the network before it gets churned out. For most simulations, the simulation is repeated with different values ranging from 1,500 to 6,000. However, when trying to distinguish between high and low churn node we used three different lifetimes to represent High, normal and low lifetime which are 1000s, 3000s and 10,000s respectively.

- **Churn Model:** For our simulation we assumed three churn generators based on the Weibull distribution. It was chosen since Weibull distribution has been shown to model real-life Peer To Peer (P2P) accurately [93].
- **Generated Traffic:** To test ONM's operation, one node generate traffic and send it to rest of the multicast group. The source node sends one 1400-bytes message every 5 seconds.
- **Network Topology:** The layout of nodes and how they connect to each other can has some effect on the results [94]. For example, having a bottleneck for the traffic might cause the buffers to overflow resulting in a low success rate. On the other hand, having unrealistically abundant resources might not cause some issues to surface. For example, assuming perfect and fast connections will not test the conditions where we have packet-drops or long delays. So, for the purpose of simulating ONM, the star-like topology was used to abstract the connections between islands. Here, each island connects to the backbone network.
- **Measurement Time:** The network is simulated for the duration of 2 hours.
- **Repetition:** Each configuration was repeated for 10 times. Then, the average of these results was taken.
- **Pastry's Parameters:** As the specific configuration is beyond the scope of this thesis, the default value set by the Pastry's main paper were chosen [95]

#### 4.2.6 *Simulation Model*

One of the most important features in OMNeT++ is its modularity design. Modules in OMNeT++ can be ether a simple module or a compound model [96]. A compound module groups other modules inside it and describe how they are connected. These models are written using the Network Description (NED) Language which is topology description language [96]. On the other hand, the simple

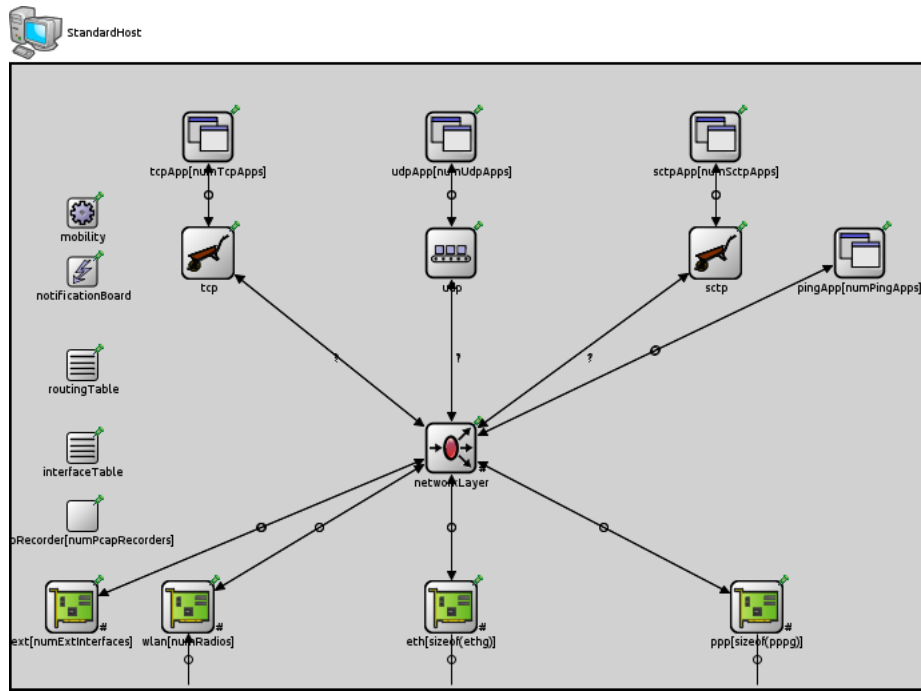


Figure 4.2: A view on the design of Standard Host component in INET

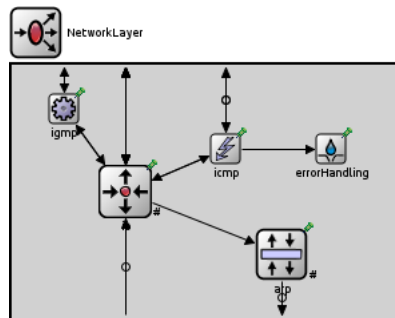


Figure 4.3: A view on the design of Standard Host component in INET

models are written in C++ and do the logic. Usually, the way these components connected reflect the OSI network model.

For example, a standard host component is a collection of other components as can be seen in figure 4.2. Each component in that figure can contain more components. So, if we take a further look at the network layer component, we will see it is just another compound module as shown in figure 4.3. This will keep going until we reach a simple component. Moreover, standard Hosts are expected to have the full OSI stack. So we can see in figure 4.2 how similar it is to the full OSI stack.

## 4.3 AMT

### 4.3.1 *Changes to The Simulation Environment*

Currently, there is no working AMT components and support in omnet++, it need to be implemented for Omnet++. The components we will try to implement and simulate are in accordance with the AMT IETF draft [44]. So, the operation and technical flow is mostly based on the this draft. Some change may be done to simplify and abstract the simulation. A justification would be given when such change happen. For this research, the aim is to have a network that consists of multiple multicast islands interconnected using AMT. After studying OMNeT++ and its available models, we will need these components to be programmed into the simulator:

**AMT GATEWAY** This component will behave as an AMT Gateway.

**AMT RELAY** This component will behave as an AMT Relay.

**AMT NETWORK MESSAGES** We will implement the different kinds of messages needed in AMT operation.

**AMT GATEWAY HOST** This host will be able to connect to AMT Relay directly.

**SIMULATION SIGNALS** We will need to add some signals to collect different relevant results.

We will study how to implement these component in this section.

#### 4.3.1.1 *AMT Gateway*

**THE MODEL COMPONENTS** To design the AMT Gateway we need to implement a range of components. The needed components can be seen in 4.4

**THE AMT-GW APP.** Most of the logic for AMT Gateway will be done in the AMT-GW App. This App will use UDP to communicate with other node in the

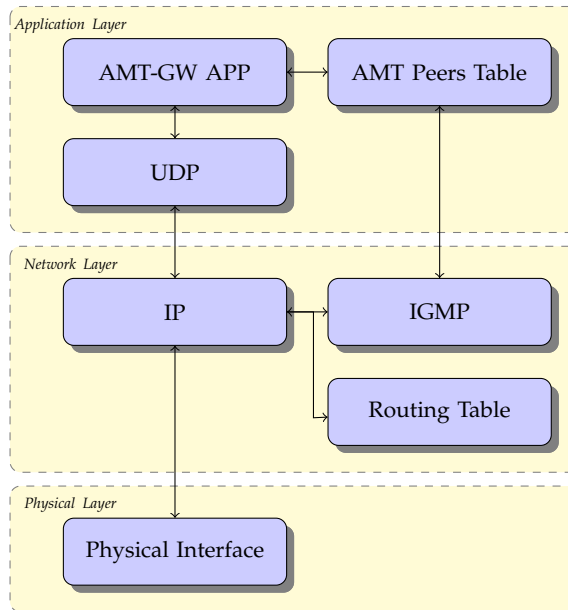


Figure 4.4: The stack of components in AMT Gateway

Peer IP	Type	Multicast Group
192.168.1.109	Host	Array{224.223.21.21}
10.2.1.12	Relay	Array{224.223.21.21}

Table 4.2: Example content of an AMT-Gateway Peer Table

network. Also, it will listen for IGMP traffic as discussed in section 2.3. The work of the App will be based on algorithm 1.

**AMT PEERS TABLE** This table will store the known peers and hosts learned using the operation of AMT. While the operation of the AMT peers table is not specified in the draft, we will implement. At this state of implementation, this table will be populated manually for each simulation. An example of this table shown in Table 4.2.

#### 4.3.1.2 AMT Relay

**THE MODEL COMPONENTS** The AMT Relay model consists of different components. These components can be seen in figure 4.5.

---

**Algorithm 1** AMT Gateway APP Algorithm

---

**Require:** msg                                   ▷ msg is the received message on any interface

- 1: **if** msg = AMT\_Relay\_Advertisement **then**
- 2:     **addRelayToAMTPeerTable** msg.source
- 3: **else if** msg = AMT\_Query & msg.group ∈ AMT\_Peer\_Table **then**
- 4:     **replay** AMT\_Membership\_Update
- 5: **else if** msg = AMT\_Data **then**
- 6:     **for all** peer ∈ AMT\_table **do**
- 7:         **forward** msg. encapsulatedData
- 8:     **end for**
- 9: **else if** msg = IGMP\_Membership\_Update **then**
- 10:     **AddHostToAMTPeerTable** msg.source
- 11: **end if**

---

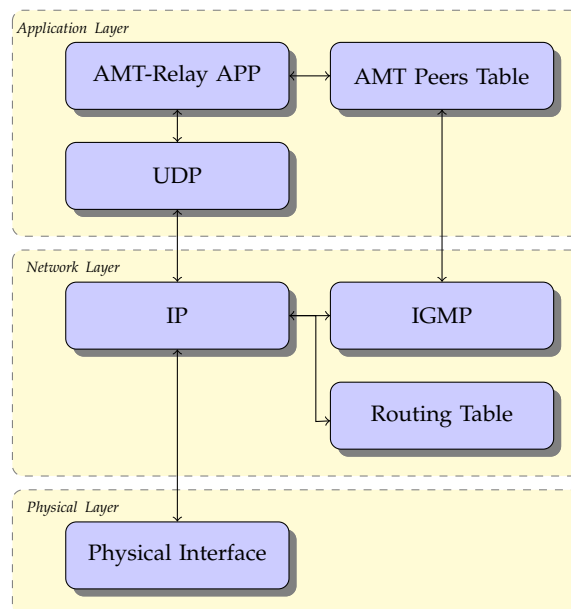


Figure 4.5: The stack of components in AMT Relay

**THE AMT-RELAY APP.** The *AMT Relay* behavior will be controlled by the AMT-Relay App that uses the UDP protocol. The App will listen and wait for a connection from an *AMT Gateway*. Also, it will listen to the local Multicast messages and forward them as appropriate. The pseudo-code that it will be used in building this App is shown in Algorithm 2.

---

**Algorithm 2** AMT Relay Component Algorithm

---

**Require:** msg ▷ msg is the received message on any interface

```

1: if msg = AMT_Relay_Discovery then
2:   replay AMT_Relay_Advertisement
3: else if msg = AMT_Request then
4:   replay AMT_Membership_Query
5: else if msg = AMT_Membership_Update then
6:   add_to_AMT_table msg.igmp.group
7: else if msg ∈ Multicast_Message then
8:   for all peer ∈ AMT_table do
9:     forward AMT_DATA.encapulate(msg)
10:  end for
11: end if

```

---

**AMT PEERS TABLE** Similar to the design of the Peer table in AMT Gateways, the Peer tables in the AMT relay will keep track of different AMT peers in the networks. However, in this table, we will only keep track of different AMT Gateways and the multicasting groups that they are interested in. At this state of implementation, this table will be populated manually for each simulation. An example of this table shown in Table 4.3.

#### 4.3.1.3 *Changes in OverSim*

Developing a new simulation to study in OverSim is very straight forward. However, since Hybrid Multicast using peer-to-peer overlay has not been implemented yet, we will need to do some coding. As has been discussed in Section 4.2.4,

Peer IP	Type	Multicast Group
10.2.1.12	Gateway	Array{224.223.21.21}
10.3.8.24	Gateway	Array{224.223.21.19}

Table 4.3: Example content of an AMT-Relay Peer Table

OverSim is very modular. We will use the existing peer-to-peer overlays available in OverSim. Moreover, we plan to implement hybrid multicasting in Oversim.

#### 4.3.1.4 Network Messages

We will define the following AMT message to be use for the negotiation the AMT tunnel:

**AMT RELAY DISCOVERY:** A discovery message from AMT Gateway to an AMT Relay.

**AMT RELAY ADVERTISEMENT:** A response from AMT Relay to the AMT Discovery message.

**AMT REQUEST:** A message from the AMT Gateway to the AMT Relay starting the *Three-way handshake* sequence.

**AMT MEMBERSHIP QUERY:** A message from the AMT Relay to the AMT Gateway encapsulating an IGMP Membership Query Message. This message is the second message in the *Three-way handshake*.

**AMT MEMBERSHIP UPDATE:** A message from the AMT Gateway to the AMT Relay. It is the last step in the *Three-way handshake*. This message will encapsulate an IGMP Membership Update Message.

**AMT MULTICAST DATA:** A message from the AMT Relay to the AMT Gateway encapsulating a multicast data message.



### 4.3.2 *The Network Model*

As a working AMT component and support is not yet available for omnet++, it need to be implemented for Omnet++. The components we will try to implement and simulate is in accordance with [1]. So, the operation and technical flow is mostly based on the this draft. Some change may be done to simplify and abstract the simulation. A justification would be given when such change happen. For this research, the aim is to have a network that consists of multiple multicast islands. These islands are interconnected using AMT.

## 4.4 BASIC IMPLEMENTATION

### 4.4.1 *Overview*

For the purpose of validating the implementation of our modules,we have build and simulated a network of interconnected multicast-enabled islands. A number of client has been scattered and assigned to islands randomly. Moreover, Each island has an AMT server which runs a generic AMT app. This app can act as either an AMT server or an AMT Gateway depending on the type of traffic received. At this stage of development most of operation of AMT is done statically. We aim to have a fully automatic AMT operation by the end of the development. At each run of this scenario, the number of multicast-enabled islands and the number of clients will be set as can be seen in figures 4.6 and 4.7. This will allow us to test our network at different sizes easily. One of the client will start generating multicast traffic and sending it to its islands. The AMT server in that

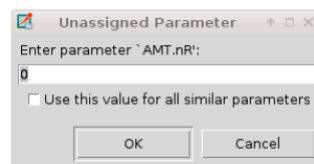


Figure 4.6: A dialog for the number of multicast islands

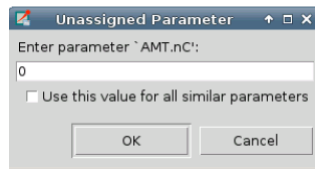


Figure 4.7: A dialog for the number of client that will be randomly scattered across different islands

island will receive the traffic and encapsulate it in a unicast UDP packets and forward them to the other AMT servers at the other islands.

#### 4.4.2 *Simulation Scenarios*

##### 4.4.2.1 *Small proof-of-concept network*

At this simulation, we wanted to verify our design. We build a network with two multicast-enabled islands as can be seen in Figure 4.8. In this network, the AMT servers will encapsulate and send any multicast message that it receives to the other island. Making the two islands behave as if they were a single multicast-enabled network. Also, we can see in Fig 4.9 a closer look on one of the AMT server. It can be seen that the UDP App is receiving AMT messages.

##### 4.4.2.2 *Applications*

AMT logic and processing is done on the AMT App. This app runs on both of AMT Gateway and AMT Relay. This app will use UDP as per the AMT draft. Also, there will be a peer table that will store different AMT addresses.

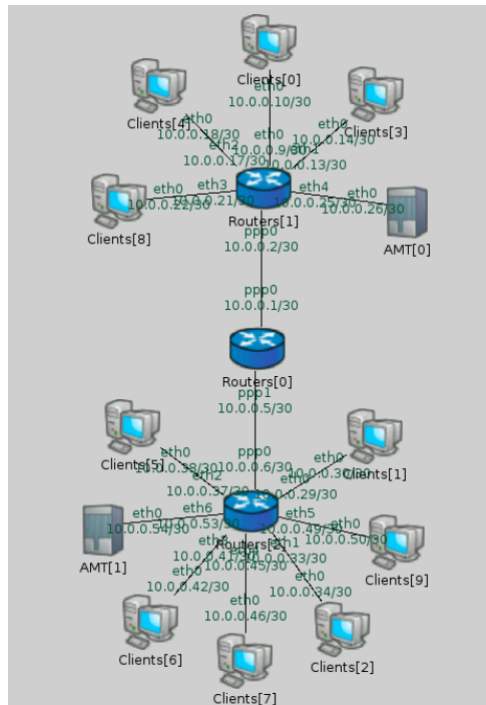


Figure 4.8: An example network with two multicast-enabled islands

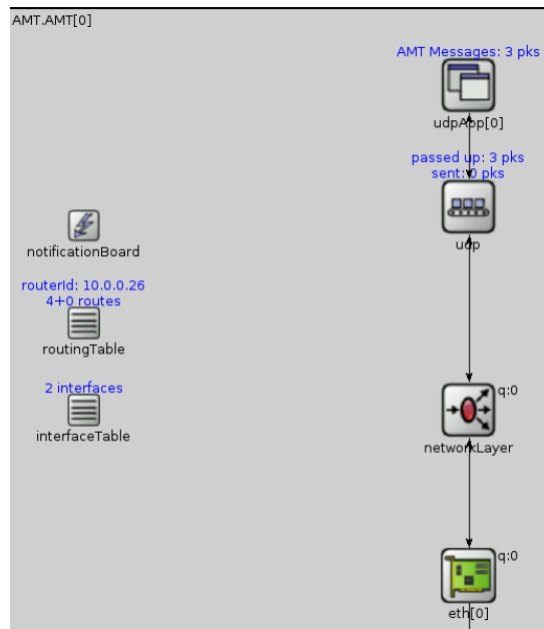


Figure 4.9: The AMT Application

## 4.5 ONM VS ALM

After the simulator was prepared, the basic operation of ONM protocol was designed and implemented . In order to test the proposed hybrid multicast framework and our implementation on Omnet++/Oversim, we have setup a number of experiments. Our experiments use a 1000 node network with a multicast group size ranging from 200 up to 1000 nodes. A source node sends multicast traffic at 1 packet every 5 seconds.

We use a setup which has 50 networks which can be used as multicast islands which are all interconnected by a backbone network of routers. We have setup the following configurations:

**ALM ONLY:** Scribe is used to manage the group and distribute the multicast data.

There is no native multicast support anywhere in the network.

**NATIVE ONLY:** IGMPv2 is used to manage the groups and distribute multicast data. The 50 networks and the backbone routers support IGMP.

**ONM:** This is our proposed model.

We have presented our approach of delivering Multicast in an environment that does not offer universal support for Native Multicast. Our approach creates an ALM tree and connects hosts that are unable to connect to the tree source natively. To optimise the performance of the delivery, the nodes that are present in the same multicast-enabled island will elect one of the nodes to act as an AMT device allowing other nodes to utilise native multicast to propagate the information inside the island. Clearly, this will decrease the number of copies of the same message to cross the backbone.

For the purpose of evaluation, we have identified five metrics: Stress, Stretch, intra-island traffic, inter-island traffic and delivery rate. Our approach has shown better results than pure ALM in every metric. However, as can be expected, the performance decreases with the increase of the number of islands. In the extreme case, if the number of islands becomes too large, ONM will yield similar results as ALM (every node would form its own single-node island).

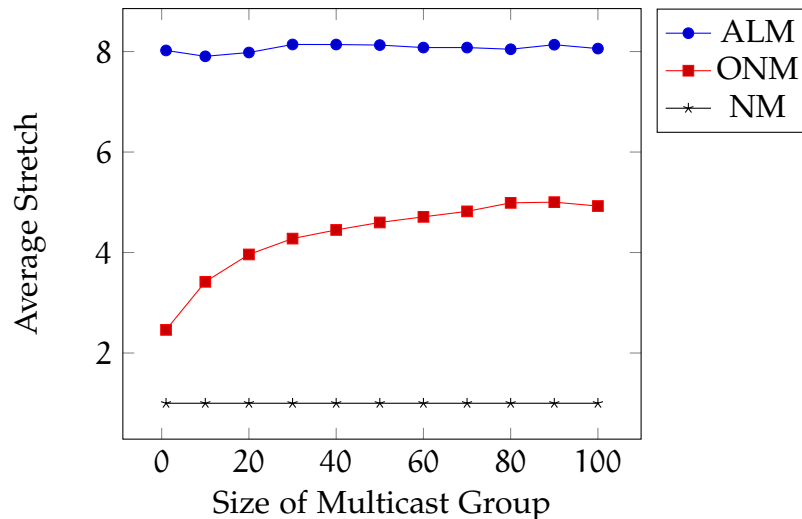


Figure 4.10: Comparing the Stretch of different multicast approaches

These results were published in the IEEE 20th International Workshop on Computer Aided Modelling and Design of Communication Links and Networks (CAMAD) [97].

The stretch of the multicast message depends heavily on the number of the nodes participating on the overlay. In the case of the pure ALM, there is no difference on the number of islands the nodes are divided into. So, we expect the ALM to be not affected by the number of islands. However, in the case of ONM, the size of the ALM tree, and subsequently the stretch, depends on how many islands exist in the topology. So, we expect the stretch to increase as more islands exist.

In Figure 4.10, with a small number of islands, we can see that ONM results in a better stretch in the overlay. However, we can see that the stretch has little dependence on the number of islands in the case of pure ALM. After a certain threshold, the stretch of ONM would converge to give similar results to ALM's stretch.

From Figure 4.11, we can see that the ALM stress on the backbone is exponentially greater than ONM's. This is due to fewer copies sent between islands. In the case of ALM, we have a linear correlation between the number of receivers and the stress on the backbone. However, ONM correlates to the number of islands. ONM results in much lower stress on the backbone.

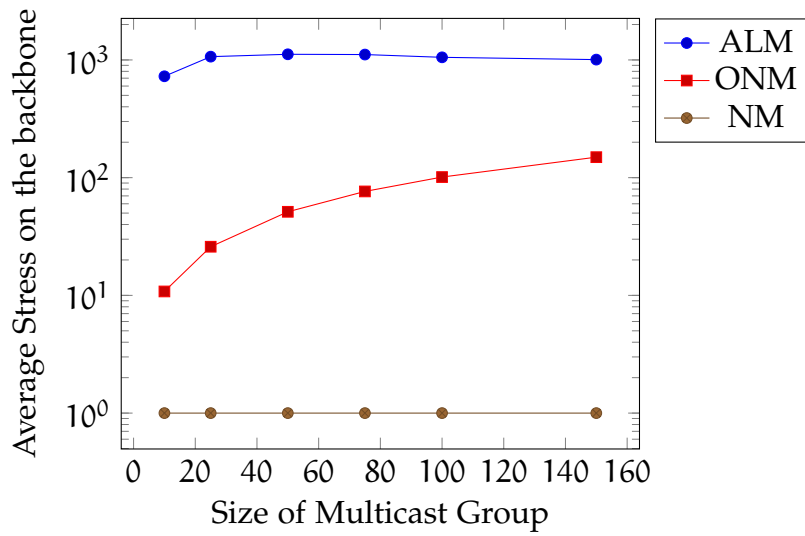


Figure 4.11: Comparing the Stress of different multicast approaches

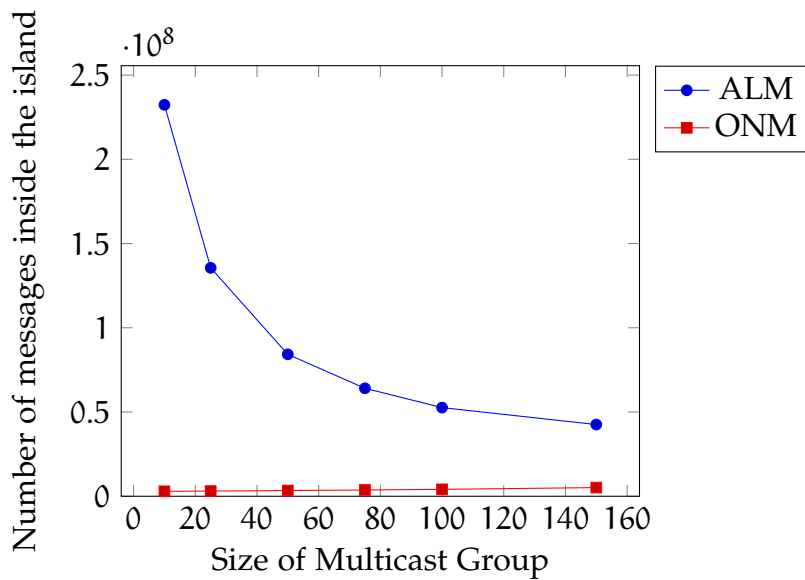


Figure 4.12: Comparing the traffic generated in each island for different multicast approach

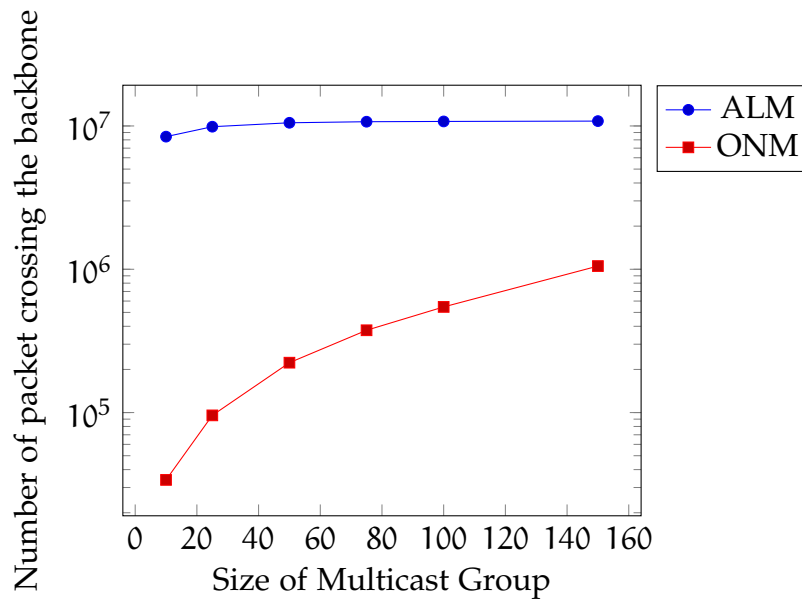


Figure 4.13: Comparing the traffic crossing the backbone for different multicast approach

Figure 4.12 display the number of packet routed inside every island. With a fixed number of nodes such as our case, as the number of islands increases, the average number of nodes per island decreases. This would result in a lower traffic inside the network. Moreover, we can see that ONM produce significantly less traffic inside the island compared with ALM.

In Figure 4.13, the traffic crossing the backbone is high regardless of the number of islands in the case of ALM. However, in the ONM case, the traffic on the backbone increases with the increase of the number of islands and is much lower than ALM.

Figure 4.14 shows the effect of using different multicasting approaches on the average delay of the multicasted messages. It can be seen in the figure that ALM has the highest delay regardless of the size of multicast group. On the other hand, Native Multicast (NM) has the lowest average delay which does not show a significant increase as the multicast group increases in size. Also, using Opportunistic Native Multicast (ONM) has resulted in a delay better than ALM and closer to Native Multicast (NM). As the size of multicast group increases, the number of peers in the overlay will increases since more islands results in more Primary Nodes.

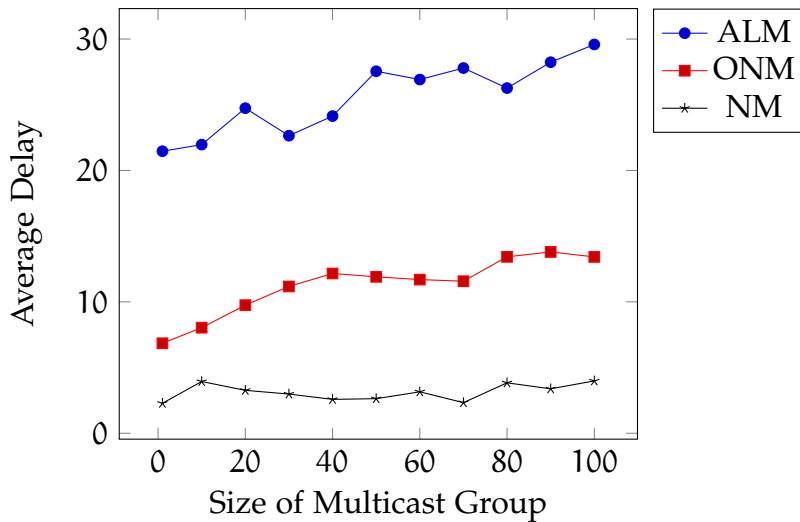


Figure 4.14: Comparing the delay for different multicast approaches

## 4.6 NUMBER OF SECONDARIES

### 4.6.1 Introduction

We have run simulations of our protocol in three different configurations. Firstly, we configured each island to select a Primary Node only. In this configuration, no secondary nodes are used. In the second configuration, the Primary Node selects a single Secondary Node. In the third configuration, the Primary Node selects two Secondary Nodes: Secondary Node 1 and Secondary Node 2. The Primary Node communicates with the Secondary Node 1 more frequently than with Secondary Node 2. The reason to select two secondary nodes, is to reduce the need for running a full election sequence when the Primary Node fails. Such a re-election is costly in terms of time and bandwidth and thus should be avoided as much as possible. Crucially, during re-election, the island is disconnected from the overlay. Consequently, any such period should be minimised.

We have simulated these three different configurations with different node churn values (lifetime) and different heartbeat intervals. Figure 4.15 depicts the results, Graph a) with a Primary node only, Graph b) with a single Secondary node, and Graph c) with two Secondary nodes. Graph a indicates that using only



the Primary Node with no Secondary Node makes the islands very susceptible to node churn and consequently disconnect from the main multicast tree. This is especially noticeable with a low heartbeat frequency. When we introduce a Secondary Node as in Graph b, we can see that the network is considerably more resilient to high node churn. Whereas with only a primary node the best possible performance was about 80% at the highest node churn, with a Secondary node 95%+ can be achieved. At the lower levels of churn the difference is less pronounced but still evident. Graph c depicts the system's performance when two Secondary Nodes are employed. Compared with the performance with a single Secondary node, there is only a slight improvement in the performance. However, the difference is more noticeable with higher churn and lower heartbeat frequency.

To analyse the full effect of the use of Secondary Nodes, the achieved message delivery and the control overhead were investigated. While the use of another special node that communicates with the Primary Node intuitively would mean added control traffic to the network, it is evident from Figure 4.16 that in fact it results in a net decrease in the overhead traffic in the island.

This seemingly counter intuitive result is due to the fact that the use of a secondary node improves the stability of the network and thus results in fewer costly re-election events. In other words, the slight increase in control traffic is offset with a reduction in election traffic.

To better observe the benefits of choosing two secondaries on the success rate, we need to focus on the environment where this configuration can be utilised the most i.e. with high churn and long heartbeat intervals. In Figure 4.17, the three configurations are combined in one graph. A low heartbeat frequency of 10 seconds was configured. It can be seen that the effects of choosing two secondaries is only noticeable up to a maximum node life time of 3000s.

However, the measures for the same environment, as shown in Figure 4.18, indicate a reduction in the overhead with an increase of the average lifetime of nodes. Figures 4.17 and 4.18 show that the use of an additional Secondary Node

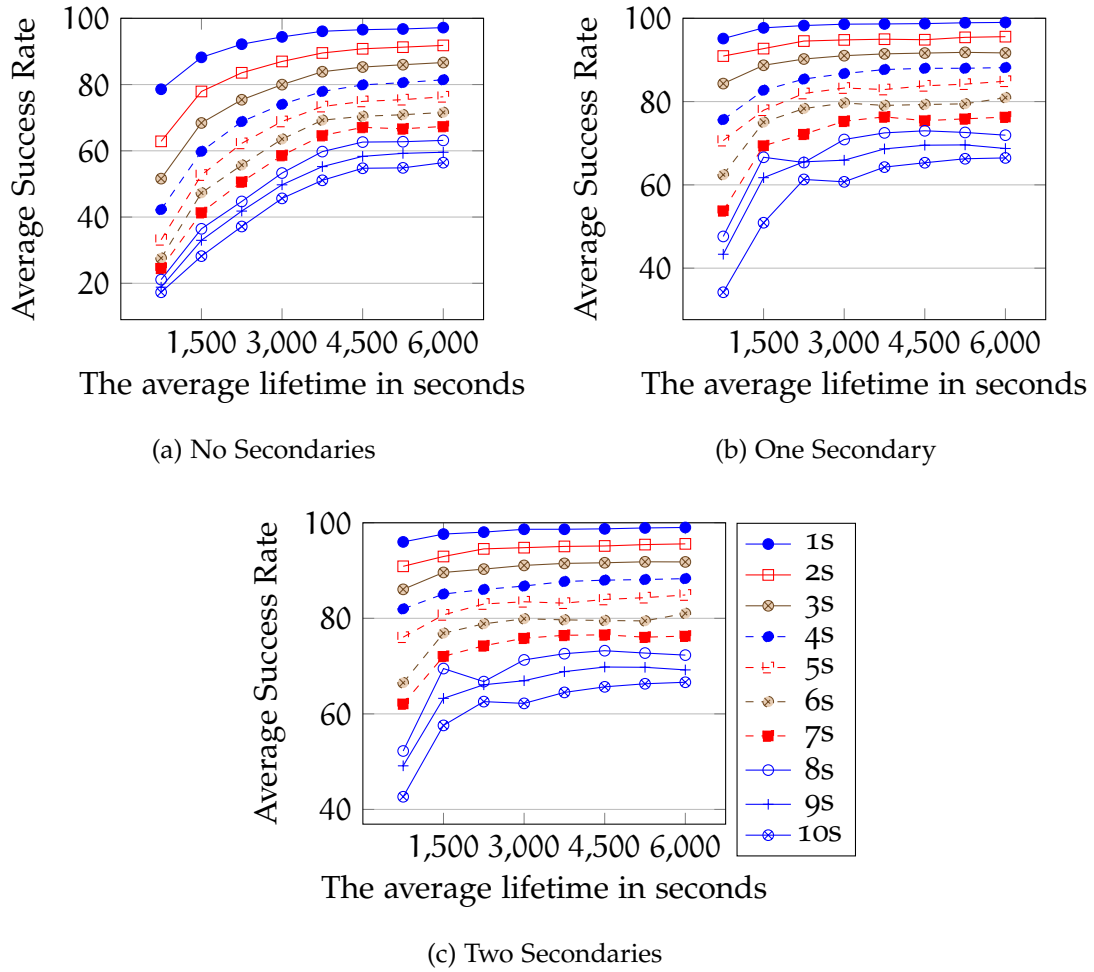


Figure 4.15: The Effect of lifetime and heartbeat intervals on Success Rate

in stable networks does not improve the success rate but incurring increased overhead.

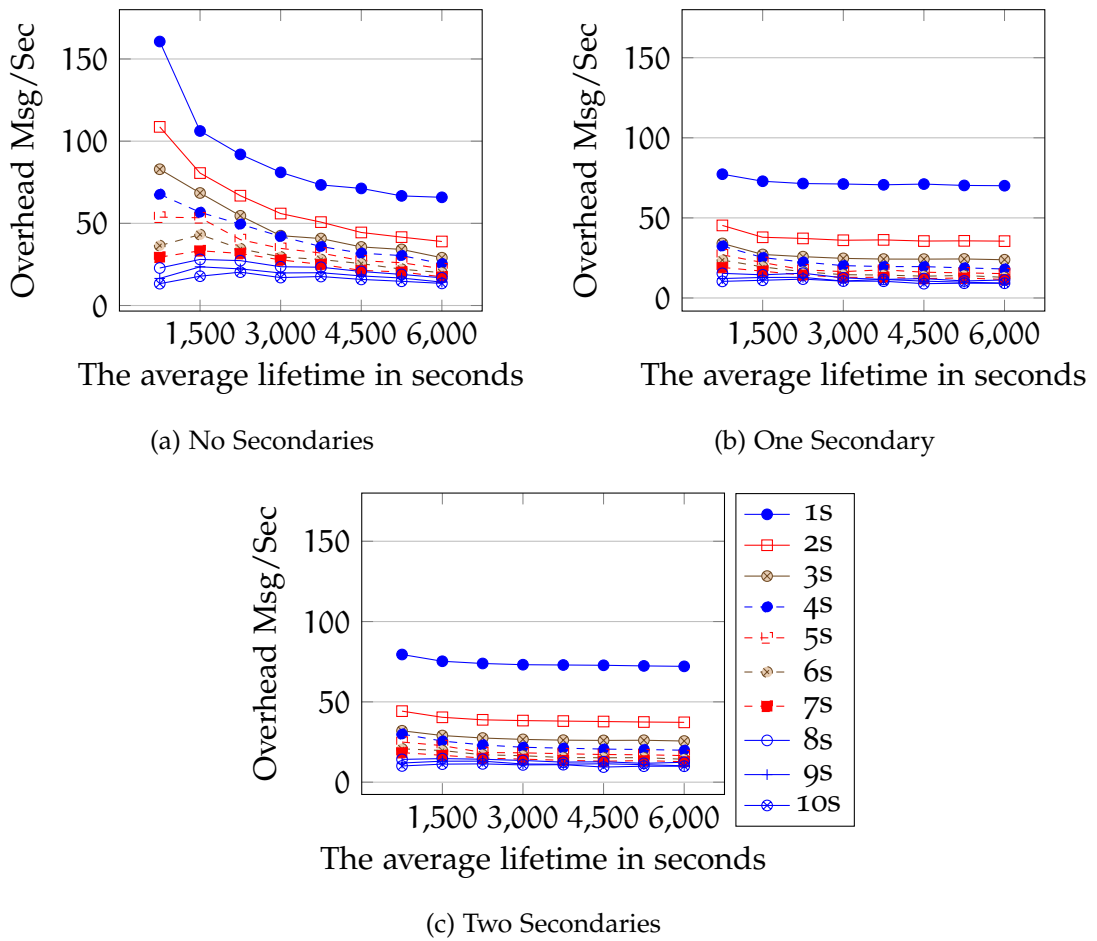


Figure 4.16: The Effect of lifetime and heartbeat intervals on Overhead

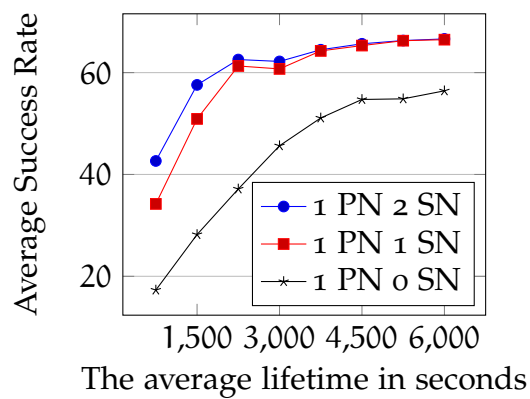


Figure 4.17: The Effect of the Number of Secondary Nodes on Success Rate (With Heartbeat = 10)

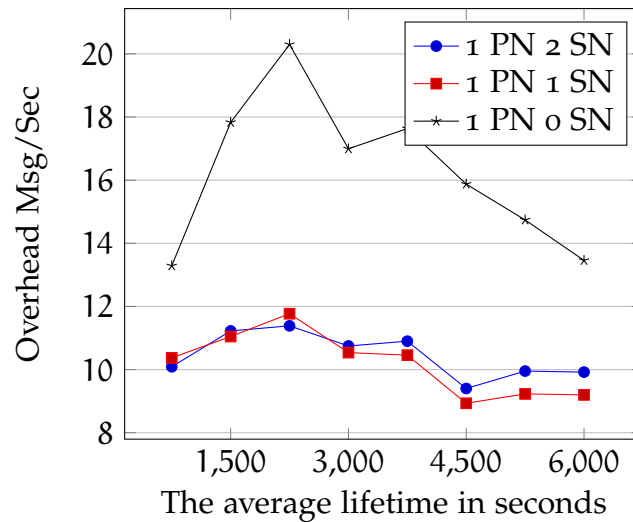


Figure 4.18: The Effect of the Number of Secondary Nodes on Overhead (With Heartbeat = 10)

#### 4.7 APPROACHES TO SELECT PRIMARY AND SECONDARY NODES

##### 4.7.1 Heterogeneity

So far our experiments have assumed that the values for the life-time of all the nodes in the network are drawn from the same probability distribution. However, it is far more likely in real networks, that different types of devices are connected to the overlay. For example, mobile peers connected to the island will have a lower life-time average than fixed devices installed by network administrators. As we have discussed earlier, selecting nodes with lower lifetime expectancy to act as primary nodes will lead to lower multicast success rates. This is due to the island being disconnected from the rest of the main multicast tree until another Primary Node has been elected and takes over. Clearly in order to achieve the best possible results, nodes that are expected to have a lower chance of churning out should be selected as primary (and secondary) nodes.

To simulate a network with nodes with different churn rates, three types of devices have been assumed to be present in our network:

- **Infrastructure:** Those are nodes who have a low likelihood of churning out of the network. However, typically there are very few of them present in the network. An example of this type of nodes are servers and routers.
- **Common:** These are the most common nodes in the network. They have an average churn rate. For example, Personal Computers fall into this category.
- **Short-lived:** These are nodes which are not very reliable and are very likely to churn out. For example, mobile devices are part of this group.

#### 4.7.2 *Distinguishing Low Churn Nodes*

In the previous set of results, nodes were chosen randomly to serve as Primary and Secondary nodes. Consequently, the islands do not necessarily utilise nodes with the longest lifetime. To allow islands to select better nodes, we have identified three possibility:

- **Manual Priority:** The network administrators can assign a priority value to nodes. Nodes with higher priority value will be selected. This method requires manual configuration of nodes but is expected to yield the best results since the expected lifetime of every node is known in advance. Using this method provides the following advantages:
  - It will allow islands to quickly and deterministically use nodes chosen by the network administrators.
  - Network administrators are able to include other factors such as security, bandwidth and processing power in their consideration.

However, this approach has the following significant drawbacks:

- It requires manual configuration of each node.
- It requires manual reconfiguration when circumstances change.
- It requires that the network administrators have control over nodes.

- It requires that the network administrators know the expected lifetime of the nodes.
- It is unable to handle events where nodes are not already classified by the network administrators.

This approach is not practical but has been included as a baseline for comparison purposes as it will yield the best possible results.

- **Passive Priority:** As the simulation time of the network increases, the probability that a low-churn node is elected will increase. This method depends on the concept of survival of the fittest. Since active nodes with a low lifetime will be churn out quickly while long lifetime nodes will last longer as active nodes. Eventually, the island will select a low churn node. When that happens, the new chosen node will have a higher probability to last longer.

When secondary nodes are introduced, the primary node will look for another node to act as secondary. This will allow to use the same process of going through different possible nodes until another node with a low-churn rate is selected. This will allow the network to minimise the need for a new re-election.

- **Age-based Priority:** Here, the oldest node will be selected. As nodes ages it will be more probable that they belong to the distribution with the highest mean life-time. We assumed that the longer the network is left running the better results that we will get.

In the following each of the three approaches is discussed and evaluated according to its strengths and weaknesses.

To test the effectiveness of these approaches, we simulated a network with 10 islands and 500 participating nodes. For our simulation we assumed three churn generators based on the Weibull distribution. The lifetime of these nodes provided in the following table:

Node Type	Percent	Lifetime Mean
Short-lived	10%	1000s
Normal	80%	3000s
Infrastructure	10%	10,000s

We configured the nodes' priority to reflect the nodes' lifetime expectancy. As before, we varied the frequency at which the nodes are communicating by changing the heartbeat intervals from 1s to 10s. We have run the simulation for different simulation times ranging from 1800s to 7200s. The simulation was carried out with 5 repetitions.

#### 4.7.3 *Manual Priority*

For manual selection of the primary node, the results are shown in Figure 4.19. The three subfigures represent having no secondary, one secondary and two secondary nodes respectively. In subfigure a), we get a success rate of 90%+ with heartbeats interval less than 5 seconds. However, with longer intervals the success rate drops to below 80%. With one Secondary Node present, as can be seen in subfigure b), there is a considerable improvement especially with longer simulation times. Slight improvement can be noted with the introduction of another Secondary Node as can be seen in subfigure c).

Figure 4.20 shows the native overhead in the island. Clearly, the heartbeat interval is a crucial influencing factor. Extending the interval from 1 second to 3 seconds cuts the native traffic down to a third. Furthermore, adding a further Secondary Node increases the control traffic in the islands only marginally. However, as before, there are no noticeable improvements in the success rate when doing so.

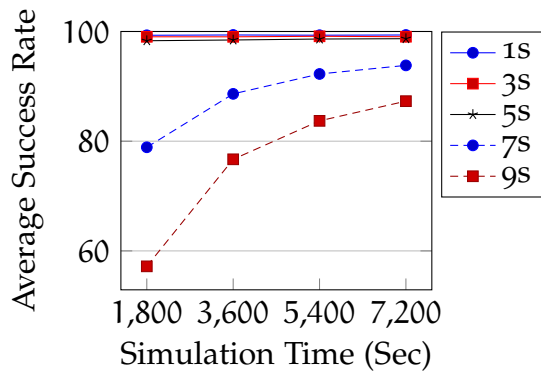
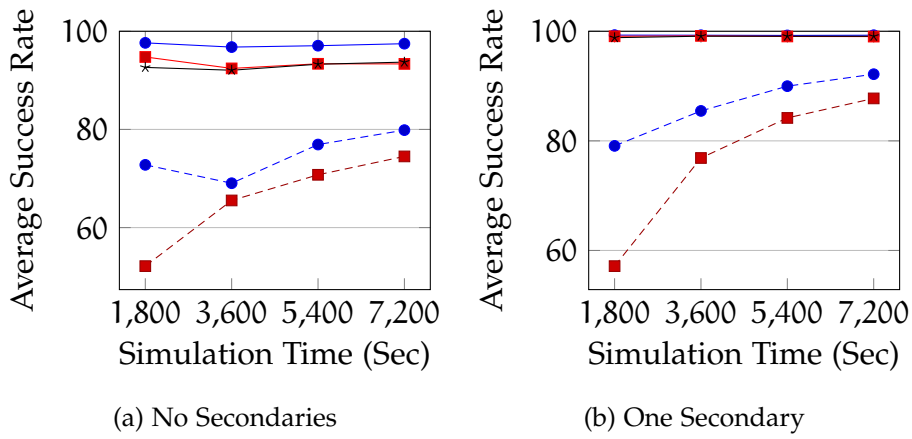


Figure 4.19: The Effect of manual selecting of Primary Nodes Success rate



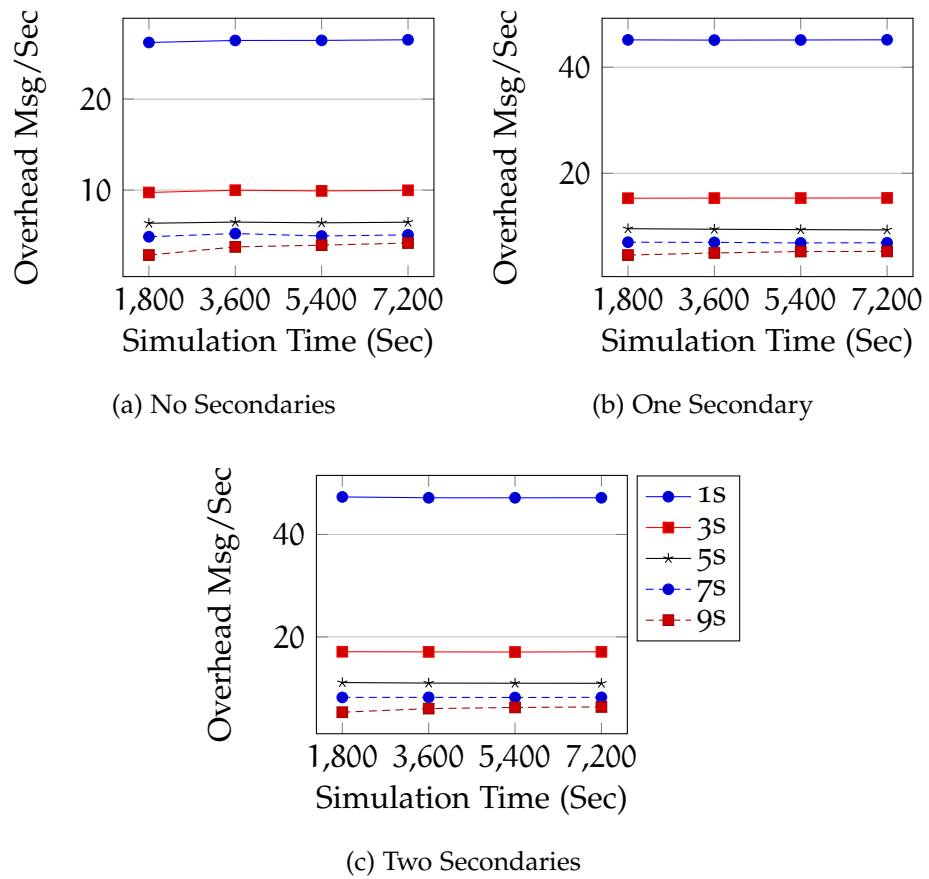


Figure 4.20: The Effect of manual selecting of Primary Nodes on Overhead

#### 4.7.4 *Passive Priority*

Figure 4.21 depicts the results for the Passive Priority approach. As can be seen in subfigure a), the network performance shows very little improvement as the simulation time increases. However, improvements are more noticeable with longer intervals between heartbeats. With a 9s heartbeat interval, the success rate increases from 30% to more than 50% across the simulation times.

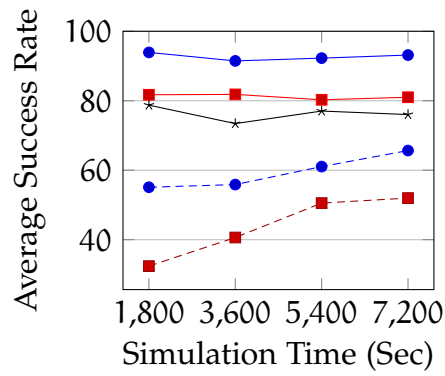
In subfigure b), the results are considerable better. For the shorter heartbeat intervals a performance of 90%+ is achieved. Even with longer heartbeat intervals a performance of 80%+ can be achieved in longer simulation runs. Subfigure c) shows very similar results as in b).

Figure 4.22 shows the native overhead in the island. Again the heartbeat interval is a deciding factor, and an additional Secondary Node has little impact on the overheads as well success rates.

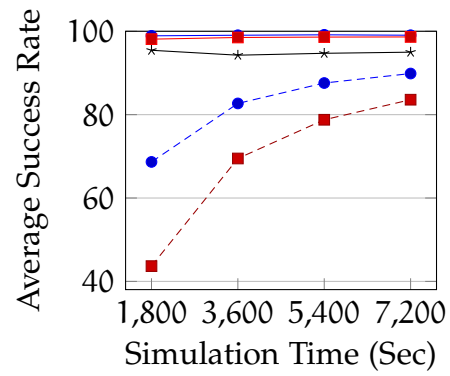
#### 4.7.5 *Age-Based Priority*

Figure 4.23 depicts the results for the Aged-based Priority approach. In subfigure a), we can see that having heartbeat intervals of 3s or less will result in a success rate of higher than 80%. With the introduction of one or two Secondary Nodes as shown in subfigures b) and c) respectively, we achieve more than 80% success in the scenarios with the longest simulation time of 7200s. However, the performance can drop considerably for longer intervals between heartbeats, and shorter simulation times.

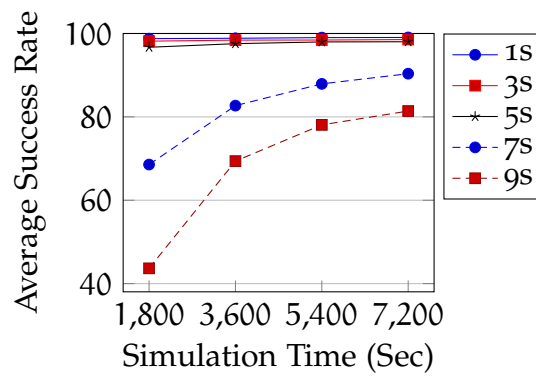
Figure 4.24 shows the overhead in the island. The results are very comparable with the previous approaches by heavily influenced by the heartbeat interval. Using one Secondary Node improves the performance, but adding additional Secondary Nodes does not yield any benefit.



(a) No Secondaries

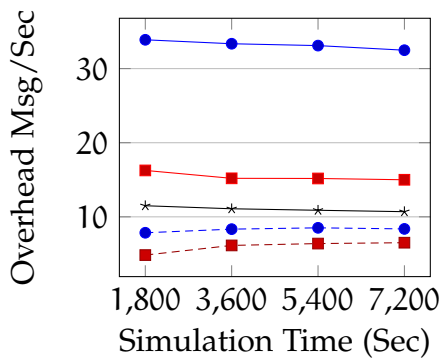


(b) One Secondary

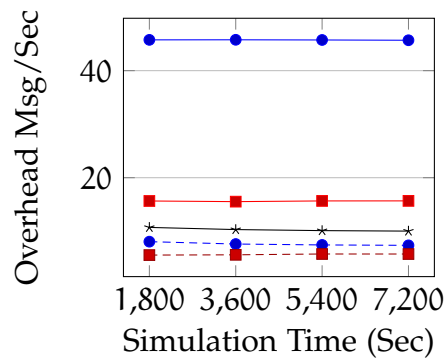


(c) Two Secondaries

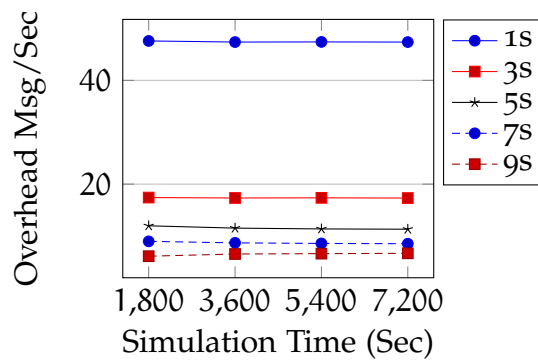
Figure 4.21: The Effect of the Length of the simulation time on Success rate



(a) No Secondaries

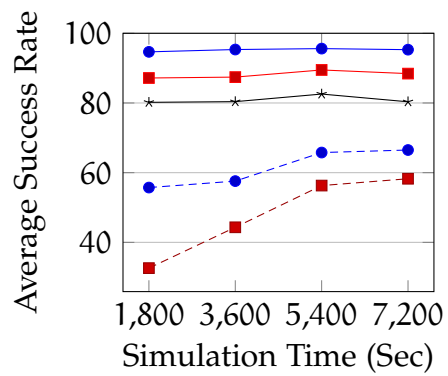


(b) One Secondary

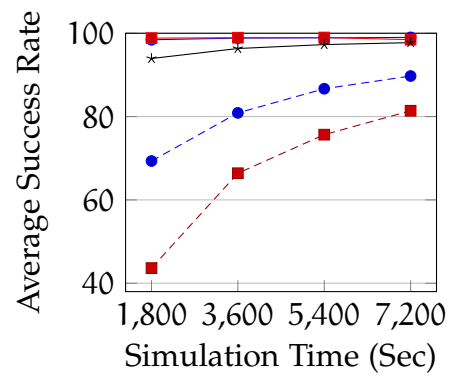


(c) Two Secondaries

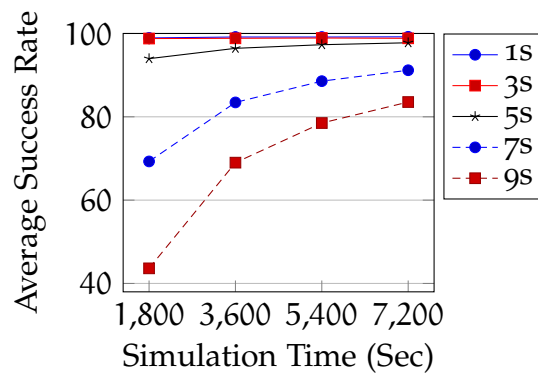
Figure 4.22: The Effect of the Length of the simulation time on Overhead



(a) No Secondaries

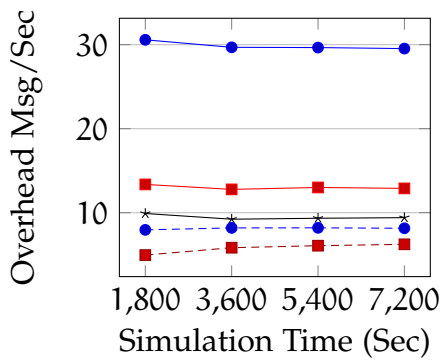


(b) One Secondary

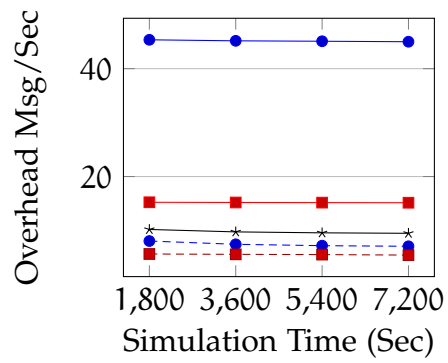


(c) Two Secondaries

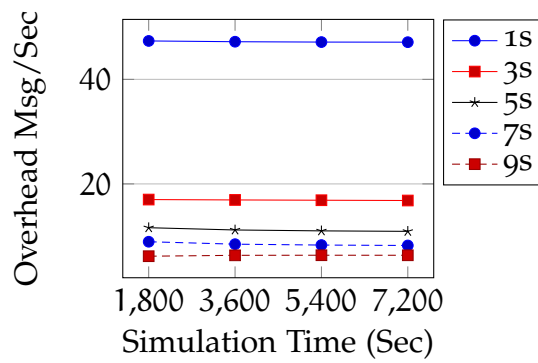
Figure 4.23: The Effect of factoring age on Success rate



(a) No Secondaries



(b) One Secondary



(c) Two Secondaries

Figure 4.24: The Effect of Factoring Age on Overhead

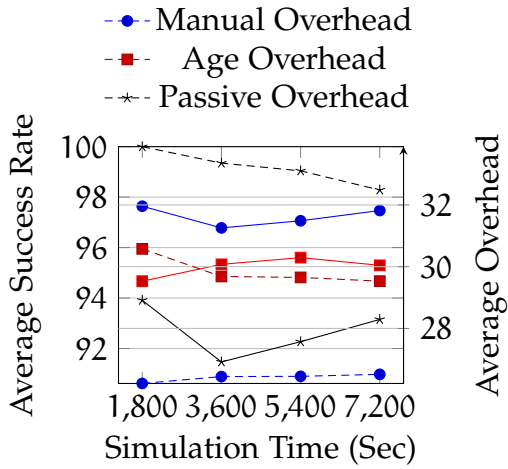
#### 4.7.6 *Comparing Types of Priority*

In the previous sections, we presented simulation results showing the effect of different types of selecting the Primary node. To aid comparison across the three approaches we have plotted selected results in graphs showing results for all three approaches. These are depicted in Figure 4.25. As can be expected the manual selection achieves the best results. However, practically, this approach cannot be employed and is shown here as baseline. Generally, the age based priority scheme performs very slightly better than the passive priority approach. However, the difference is rather minor.

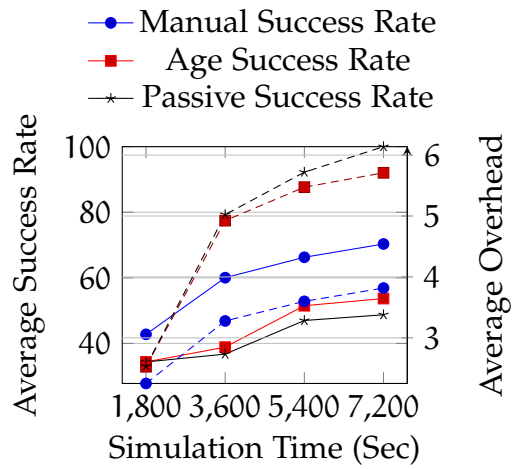
More specifically, with the higher heartbeat frequency, as in subfigures a), c) and e), the difference between the three types is especially low. This is due to the fact that with the high heartbeat frequency, the islands can detect and react to failure very quickly. This in turn results in the down-time suffered through the election process very small. Thus, used to select a primary node does not have a big effect.

Also, the number of Secondary Nodes can effect the success rate especially with low heartbeat frequency as in Figures a, c and e.

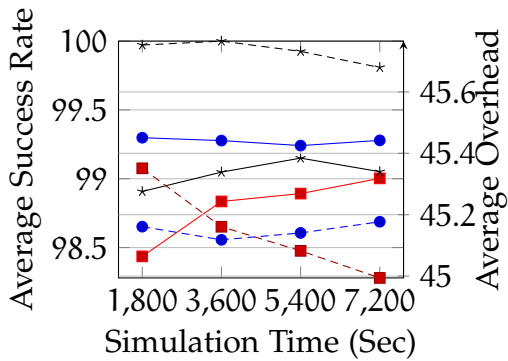
In addition, the control overhead in islands is shown for each setup. In conclusion, for environments where the island can react quickly to changes, as in a), c) and e), the use of Secondary Nodes will only increase the overhead without noticeable increase in the success rate. However, with slower reacting islands as in b), d) and f), the use of Secondary Nodes decreases the control traffic as it will avoid costly re-elections.



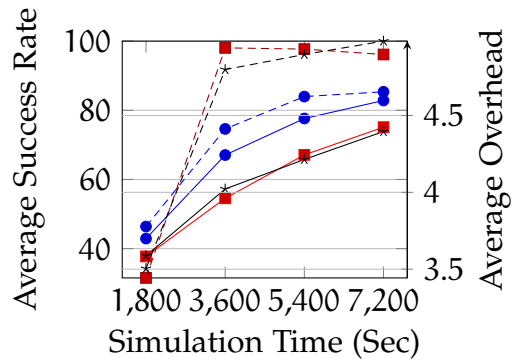
(a) HB = 1 with No Secondaries



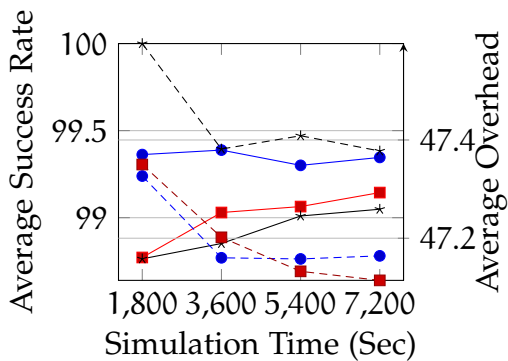
(b) HB = 10 with No Secondaries



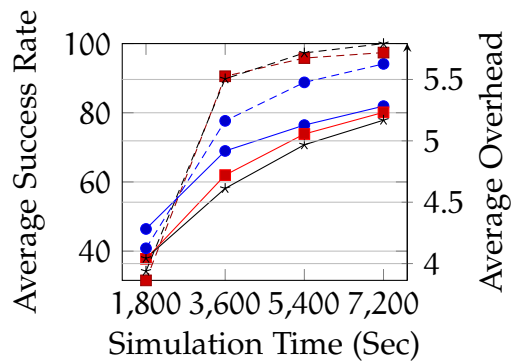
(c) HB = 1 with One Secondary



(d) HB = 10 with One Secondary



(e) HB = 1 with Two Secondaries



(f) HB = 10 with Two Secondaries

Figure 4.25: The Effect of different methods of selecting the Primary Nodes with different heartbeat frequencies and number of Secondaries



## 4.8 DYNAMIC HEARTBEAT INTERVALS

For the islands to detect and react to changes in the network, exchanges of heartbeat messages between nodes need to take place. Setting the frequency too high will result in unnecessary control traffic and unnecessary overhead. On the other hand, setting it too low will delay reacting to failures and will result in lower success rates.

The optimum value of the heartbeat interval depends on the network condition. If the network is stable, a lower frequency will be sufficient as it lowers the overhead. However, in an unstable network, a higher frequency is required since it will allow the network to react faster to changes which are more likely in such networks.

### 4.8.1 *Dynamic Control Interval*

Until now, our simulations assumed that the heartbeat interval is fixed throughout the simulation. In this section we propose an approach to let the network change this value depending on the network conditions. This allows for a more automated and scalable operation of the network. The approach proposes that the interval between heartbeats be changed from an initial value  $x_i$  to a final value  $x_f$ . The change from  $x_i$  to  $x_f$  will take a set amount of time  $T_{\text{Transition}}$ . The change can be implemented using two different methods which we will discuss below: Probation Period and Graduate Trust. The two methods achieve the same value of  $x_f$  in the same amount of time  $T_{\text{Transition}}$ .

**PROBATION PERIOD:** This approach assumes that newly selected nodes are "high-risk" until some probation time has passed. Then, the heartbeat frequency will be decreased. Once a newly selected Secondary Node has been available for an initial period, The Primary node increase its trust in that node's operation and decreases the frequency at which the heartbeats are exchanged.

GRADUATE TRUST: Here, the Primary Node starts with a cautious value for the heartbeat frequency. This value will be relaxed with each successful heartbeat acknowledgement. This process will continue until a threshold is reached. In this approach, the primary node will start with the initial heartbeat interval  $x_i$  and with each heartbeat acknowledgement, the primary node will increase the interval by a fixed increment  $c$ . This process will continue until a threshold is reached  $x_f$ . We can calculate the value of  $x_f$  by using this equation:

$$x_f = x_i + rc \quad (4.1)$$

Where  $r$  is the number of heartbeats exchanges in the transition time.

To test the effectiveness of this approach, a network with 10 islands and 500 participating nodes has been simulated. The lifetime of these nodes is distributed according to the following table:

Node Type	Percent	Lifetime Mean
Short-lived	10%	1000s
Normal	80%	3000s
Infrastructure	10%	10,000s

For comparison networks with static heartbeat intervals ranging from 1s to 10s between heartbeats have been simulated. The simulations were run with 10 repetitions. For the following simulations,  $x_i$  was varied between 1 and 10. The values for  $x_f$  and  $T_{\text{Transition}}$  were chosen according to Table 4.4.

#### 4.8.2 Probation Period

As can be seen from Figure 4.26 subfigure a), both static and dynamic approaches achieve a very similar success rate. However, we can see in subfigure b) a decrease in the overhead while not impacting on the success rate. In the same graph, the effect of the probation time is noticeable, in that a short probation time decreases the message overhead. For example, with heartbeat intervals equal to 1 second, choosing 500 seconds instead of 1000 seconds as the probation period, the message

Table 4.4: Parameters for  $T_{\text{Transition}}$ 

Type	$x_f$	$T_{\text{Transition}}$
Probation Time	$2 \times x_i$	500S
Probation Time	$2 \times x_i$	1000S
Probation Time	$4 \times x_i$	500S
Probation Time	$4 \times x_i$	1000S
Graduate Start	$2 \times x_i$	500S
Graduate Start	$2 \times x_i$	1000S
Graduate Start	$4 \times x_i$	500S
Graduate Start	$4 \times x_i$	1000S

overhead is reduced by about 30%. As is to be expected, the effect becomes less pronounced with lower heartbeat frequencies.

### 4.8.3 Graduate Trust

Figure 4.27 subfigure a) shows that both static and dynamic approaches have very similar success rate, albeit with a marginally wider spread than with the Probation Period approach. Figure 4.27 subfigure b) demonstrates a reduction in message overhead which even exceeds the reduction achieved through the Probation Period approach. Unlike with the probation period method, the value of  $T_{\text{transition}}$  had less of an effect on the number of heartbeat message exchanged. For exempling, changing  $T_{\text{transition}}$  from 1000 to 500 decreased the number of heartbeat messages by only around 10%.

In order to be able to compare the two approaches, the two networks must take the same amount of time to reach the final state  $T_{\text{transition}}$ . With the probation period method, this is achieved by setting the probation value to the desired time. However, using the Graduate Start method, the increment  $C$  can be used.

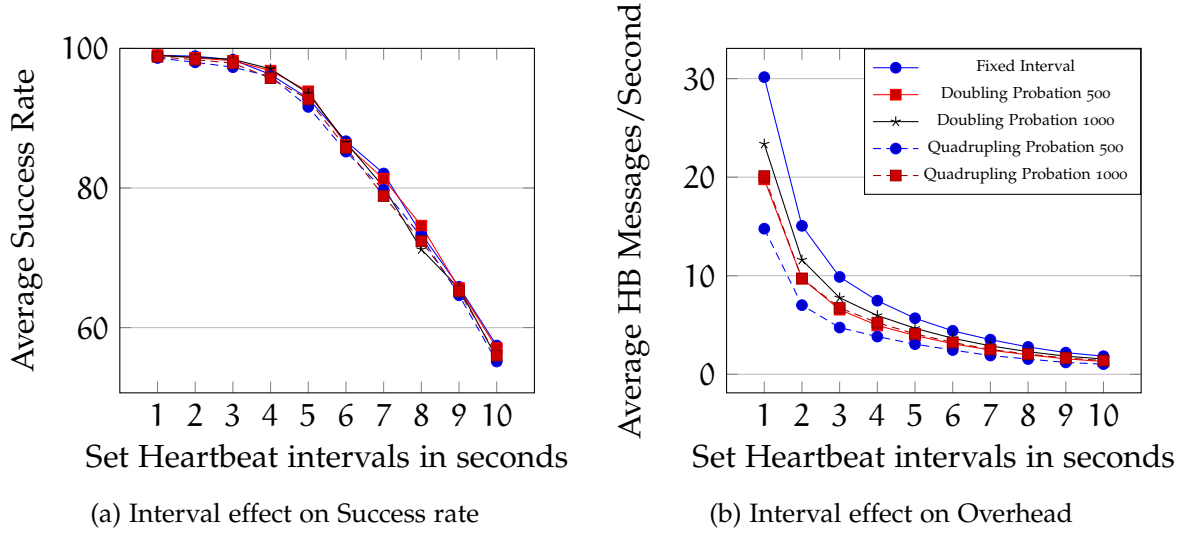


Figure 4.26: The Effect of Probation-time Dynamic Interval

To calculate the  $T_{\text{transition}}$  of the Graduate Start method the following calculation can be used:

$$\begin{aligned}
 T_{\text{transition}} &= \sum_{k=0}^r (x_i + kc) \\
 &= (r+1)x_i + \sum_{k=1}^r kc \\
 &= (r+1)x_i + \frac{rc(r+1)}{2} \\
 &= \frac{2(r+1)x_i + rc(r+1)}{2} \tag{4.2} \\
 &= \frac{(r+1)(2x_i + rc)}{2} \\
 &= \frac{(r+1)(x_i + (x_i + rc))}{2} \quad (\text{substituting } x_f \text{ from equation 4.1}) \\
 T_{\text{transition}} &= (r+1) \frac{x_i + x_f}{2}
 \end{aligned}$$

To find  $r$  for a specific amount of transition time, as in our case, the equation can be transformed to:

$$\begin{aligned}
 T_{\text{transition}} &= (r+1) \frac{x_i + x_f}{2} \\
 (r+1) &= T_{\text{transition}} \frac{2}{x_i + x_f} \tag{4.3} \\
 r &= \frac{2T_{\text{transition}}}{x_i + x_f} - 1
 \end{aligned}$$

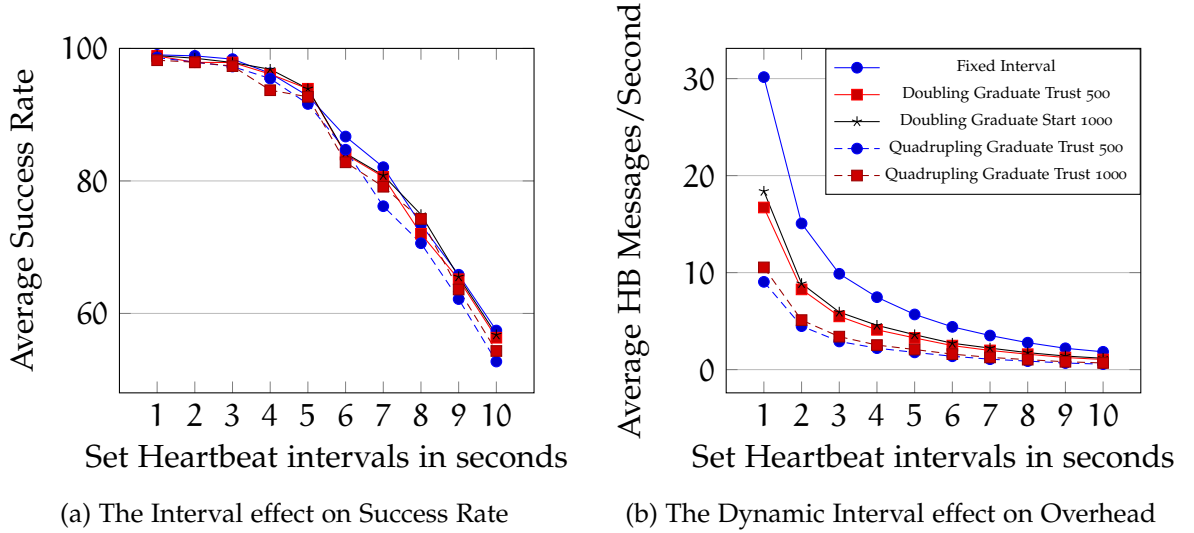


Figure 4.27: The Effect of Dynamic Interval in Graduate Trust

From Equation 4.1, the increment  $c$  can be calculated:

$$\begin{aligned}
 x_f &= x_i + rc \\
 rc &= x_f - x_i \\
 c &= \frac{x_f - x_i}{r}
 \end{aligned} \tag{4.4}$$

By combining Equation 4.4 and Equation 4.3, the following equation can be used to calculate the value of the increment  $c$ :

$$c = \frac{x_f - x_i}{\frac{2T_{\text{transition}}}{x_i + x_f} - 1} \tag{4.5}$$

For experimentation,  $T_{\text{transition}}$  values of 1000 and 500 seconds are used. The simulations included final value of heartbeat interval  $x_f$  to be double and quadruple the initial value  $x_i$ . Using Equation 4.5, the values of  $c$  can be calculated and are shown in the following table:

Table 4.5: Calculation of C

	$x_i$	$x_f$	c	
			$T_{\text{transition}} = 500$	$T_{\text{transition}} = 1000$
Doubling	1	2	0.003009027	0.001502253
	2	4	0.012072435	0.006018054
	3	6	0.027245207	0.013561025
	4	8	0.048582996	0.024144869
	5	10	0.076142132	0.037783375
	6	12	0.109979633	0.054490414
	7	14	0.150153218	0.074279939
	8	16	0.196721311	0.097165992
	9	18	0.249743063	0.123162696
	10	20	0.309278351	0.152284264
Quadrupling	1	4	0.015075377	0.007518797
	2	8	0.060606061	0.030150754
	3	12	0.137055838	0.068010076
	4	16	0.244897959	0.121212121
	5	20	0.384615385	0.189873418
	6	24	0.556701031	0.274111675
	7	28	0.761658031	0.374045802
	8	32	1	0.489795918
	9	36	1.272251309	0.621483376
	10	40	1.578947368	0.769230769

## 4.9 DETECTION OF NODE FAILURE

### 4.9.1 Introduction

The purpose of Heartbeat messages is to let nodes keep track of the availability of other nodes in the network. By detecting failures, nodes can take actions. For example, a secondary node can replace and act as a Primary nodes as soon as it detects that the Primary node is no longer available in the network.

Due to the unreliable nature of network connections, some packets may be dropped. In that case, if enough consecutive heartbeats get dropped, the node sending them may be falsely declared dead. This may lead a network to enter an unneeded reelection or make the Secondary Node takes control and declare itself as a Primary node even though the Primary node is still operating.

### 4.9.2 Heartbeat Timeout Timer

To avoid unnecessary election events as much as possible, at the Primary and the Secondary nodes, there is a timeout timer that keeps track of the exchanged heartbeats. The heartbeat timers will be reset every time a heartbeat message arrives. The timer timeout value will be  $T_{\text{HeartbeatTimeout}}$ . In our implementation,  $T_{\text{HeartbeatTimeout}}$  is set to be a multiple of  $T_{\text{heartbeat}}$  as in the following equation:

$$T_{\text{HeartbeatTimeout}} = N \times T_{\text{heartbeat}} \quad (4.6)$$

The value of  $N$  in Equation 4.6 represents the time the node will wait between heartbeats. So, a value of 1 would mean that timer will wait for the period of a single heartbeat. This will result in that any delay or jitter in the network will result in the node to be declared dead if a single heartbeat message is missed. Clearly, a value of 1 is not a advisable setting. Alternatively, a large value of  $N$  will result in the network reacting with a considerable delay to changes in the network.

### 4.9.3 *Selecting the value of N*

To select a reasonable value of N, experiments with different values for N were carried out. Two relevant statistics were collected for every run:

**UNNEEDED COUP:** This represents the number of times another node took over as Primary Node while the previous Primary Node is still operating.

**SUCCESS RATE:** The percentage of messages that were delivered successfully.

We simulated 500 nodes participating in ONM which were randomly divided into 10 islands. In the experimentation we used the following packet drop rates:

- **No Packet Drops:** In this scenario, all packets eventually reach their destination.
- **1% Packet Drops:** Here, 1% of the packets will be dropped before reaching their destination.
- **2% Packet Drops:** Here, 2% of the packets will be dropped before reaching their destination.
- **3% Packet Drops:** Here, 3% of the packets will be dropped before reaching their destination.
- **4% Packet Drops:** Here, 4% of the packets will be dropped before reaching their destination.

Each simulation run was repeated 10 times and the results are depicted in Figure 4.28.

In Figure 4.28 subfigure a), the number of unneeded takeovers are depicted with respect to different values of drop rate and heartbeat timeout multiplier N. The probability that consecutive heartbeats will be dropped can be calculated using the following equation:

$$P_{N\_consecutive\_drops} = (P_{drop})^N \quad (4.7)$$



Table 4.6: The chance of unneeded takeovers for different values of N and Packet Drops

Packet Drops	Multiplier N			
	1	2	3	4
0%	0	0	0	0
1%	0.01	0.0001	0.000001	0.00000001
2%	0.02	0.0004	0.000008	0.00000016
3%	0.03	0.0009	0.000027	0.00000081
4%	0.04	0.0016	0.000064	0.00000256

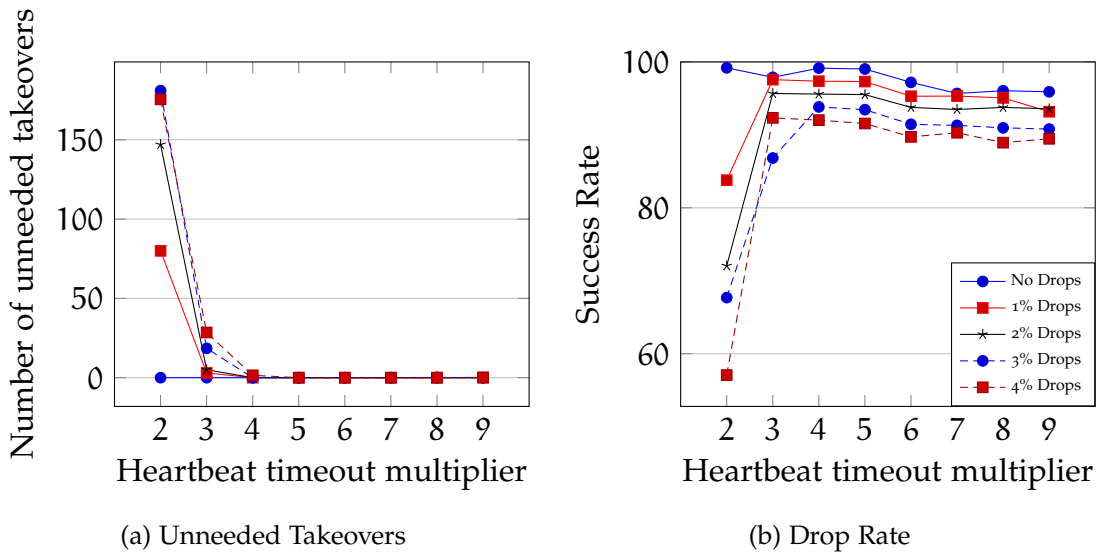


Figure 4.28: The effect of heartbeat timeout value

In Equation 4.7,  $P_{\text{drop}}$  is the probability of dropping a particular packets. Also,  $P_{N\_consecutive\_drops}$  is the chance that a N consecutive packets will be dropped. As can be seen in Figure 4.28, using 4 for N will result in hardly any unneeded takeovers. Equation 4.7 can be used to calculate the chances of unneeded takeovers and the results are provided in Table 4.6. Similarly, in Figure 4.28 subfigure b), it can be seen that the success rate starts reaches it maximum from  $N = 4$ . This indicates that there is no advantage to be gained of having values of N to be larger than 4.

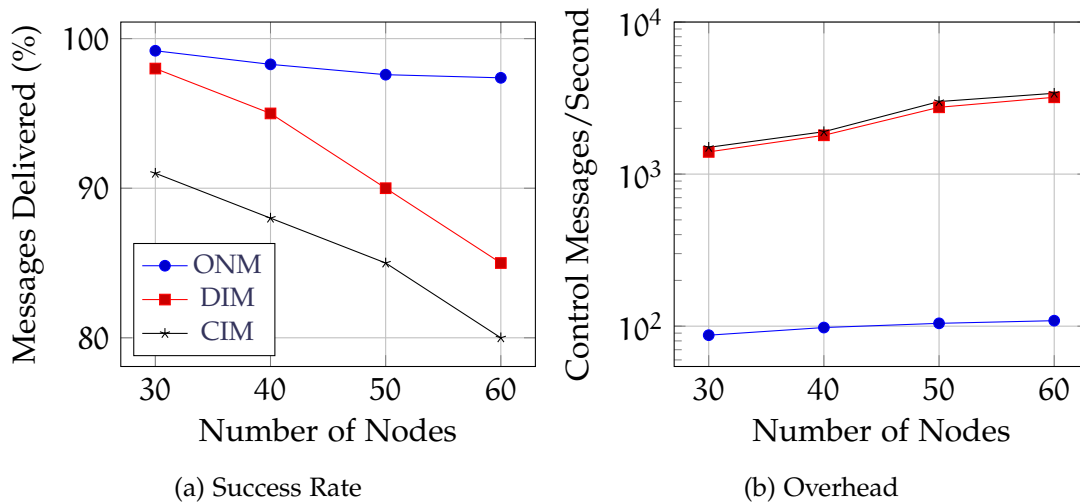


Figure 4.29: Comparing Island Multicast (IM) with Opportunistic Native Multicast (ONM)

#### 4.10 COMPARING ONM WITH IM

Similar to Opportunistic Native Multicast (ONM), IM uses unicast to connect islands while utilise Native Multicast (NM) to deliver data the island. As discussed in Section 2.1.4.2, IM can be implemented either as Centralised Island Multicast (CIM) or Distributed Island Multicast (DIM). Since CIM requires setting a central nodes manually, DIM is the one that is comparable to Opportunistic Native Multicast (ONM). The advantages of using Opportunistic Native Multicast (ONM) over Island Multicast (IM) were discussed in Section 3.1.3.

##### 4.10.1 Performance Analysis

Figure 4.29 shows the difference between Opportunistic Native Multicast (ONM) and the two types of IM: CIM and DIM. Subfigure a) depicts the performance of the three approaches with respect to message delivery. It can be seen that ONM significantly outperforms both CIM and DIM. The difference is especially pronounced with the larger network sizes. The results published for CIM and DIM only cover network sizes up to 60 nodes, at which point their performances deteriorate sharply. ONM was tested with considerably larger networks. As for

the overhead, Opportunistic Native Multicast (ONM) requires significantly less network traffic than the other two approaches. In fact the difference is more than an order of magnitude. Overall, it can be concluded that ONM outperforms both CIM and DIM on both, the message delivery and the incurred overhead.

#### 4.11 SUMMARY AND CONCLUSION

This chapter has evaluated the operation of ONM under different scenarios and parameters. First, the testing environment was discussed and a simulator was selected. Also, the needed changes were discussed. After that, a basic implementation was planned and analysed. This chapter discussed three different approaches when selecting the Primary Nodes of the islands: Manual, Passive and Age-based. Moreover, the chapter evaluated how ONM will be able to detect the stability of the network and will relax the frequency of the heartbeats messages accordingly which was shown to reduce the control overhead significantly. Two approaches were analysed and tested: Probation-time and Graduate Trust. In ONM, failure of the Primary Node is detected by missing a number of subsequent heartbeat messages. When a threshold is reached, the Primary Node is declared dead and the Secondary Node will take over. If no secondary Node exist, a reelection is triggered. A range of values for this threshold was studied and analysed under different churn rates and packet drops probabilities. The chapter also tried to find the optimal value that will not declare a Primary Node dead too soon risking unnecessary take over nor take a long time to react to the failure of the Primary Node. From the different simulations in this chapter, Tables 4.7, 4.8, 4.9, 4.10, 4.11 and 4.12 summarise the results got from simulating different ways to select the Primary Node (PN) under different configurations. Table 4.7 shows the success rate of different approaches by taking the average number of delivered messages. Table 4.8 shows the cost in overhead for each protocol. This cost is calculated by the average number of control messages per second. Table 4.9 shows the average time taken by messages to reach their destination. Table 4.10 shows the number of messages present inside the island. Table 4.11 shows the stress on the backbone

link connecting different islands. The stress is measured by the number of copies of a message crossing the same link or node. Table 4.12 shows average stretch which is the length of the path taken compared to the path taken using unicast.

Heartbeats Interval	Number of Secondaries	Average Lifetime	Method of Selecting the PN		
			Age	Manual	Passive
10	0	1800	34.35362011	42.76446197	34.35362011
10	0	3600	38.86016931	60.03750251	36.73770672
10	0	5400	51.49887989	66.26803414	46.98806018
10	0	7200	53.67884411	70.31703481	48.73963851
1	0	1800	94.66887625	97.63945153	93.911531
1	0	3600	95.33641902	96.77989771	91.4686085
1	0	5400	95.601892	97.06165166	92.27032047
1	0	7200	95.29747918	97.46859089	93.15260085
10	1	1800	37.75979373	42.90866952	37.94683838
10	1	3600	54.5500593	67.0655779	57.24941523
10	1	5400	67.10899509	77.6245473	65.78806208
10	1	7200	75.12997763	82.83228123	73.86443233
1	1	1800	98.43596918	99.29802376	98.90739958
1	1	3600	98.83510352	99.27730578	99.04805285
1	1	5400	98.89221374	99.24080928	99.14988825
1	1	7200	99.00279273	99.27880235	99.0513579
10	2	1800	37.77130778	46.40896702	37.93658411
10	2	3600	61.9725013	68.95577659	58.13087394
10	2	5400	73.85618597	76.43351446	70.73614223
10	2	7200	80.1213336	81.91667814	77.8518676
1	2	1800	98.77151717	99.36239079	98.76371693
1	2	3600	99.02975603	99.38823854	98.85182422
1	2	5400	99.06397	99.30058267	99.00974222
1	2	7200	99.14427551	99.34615242	99.04857361

Table 4.7: Success Rate of different configurations of Lifetime, Number of Secondaries and Primary Selection Method

Heartbeats Interval	Number of Secondaries	Average Lifetime	Method of Selecting the PN		
			Age	Manual	Passive
10	0	1800	2.525944444	2.251444444	2.534888889
10	0	3600	4.923944444	3.277194444	5.023277778
10	0	5400	5.47337037	3.599611111	5.719240741
10	0	7200	5.709027778	3.817722222	6.138347222
1	0	1800	30.57733333	26.21961111	33.88577778
1	0	3600	29.68358333	26.44244444	33.35347222
1	0	5400	29.65298148	26.4497963	33.10268519
1	0	7200	29.53183333	26.51515278	32.48245833
10	1	1800	3.443222222	3.777555556	3.504111111
10	1	3600	4.94	4.412722222	4.79875
10	1	5400	4.932055556	4.623259259	4.897018519
10	1	7200	4.897013889	4.65425	4.984125
1	1	1800	45.35111111	45.161	45.75272222
1	1	3600	45.16066667	45.11816667	45.76558333
1	1	5400	45.08222222	45.14057407	45.73211111
1	1	7200	44.99398611	45.17697222	45.67966667
10	2	1800	3.863166667	4.124555556	3.938388889
10	2	3600	5.526277778	5.162777778	5.505305556
10	2	5400	5.674444444	5.476425926	5.715981481
10	2	7200	5.719069444	5.626930556	5.792652778
1	2	1800	47.35005556	47.32655556	47.59627778
1	2	3600	47.20180556	47.16016667	47.38158333
1	2	5400	47.13261111	47.15783333	47.40859259
1	2	7200	47.11420833	47.16411111	47.37758333

Table 4.8: Average overhead of different configurations of Lifetime, Number of Secondaries and Primary Selection Method

Number of Islands	Multicasting Technology		
	ALM	ONM	NM
1	21.46383451	6.84649357	2.280723666
10	21.96618269	8.03415829	3.935657726
20	24.7402539	9.74770814	3.266022472
30	22.64443392	11.17036953	2.982579716
40	24.13815344	12.15810212	2.57746619
50	27.54920938	11.90544394	2.631426154
60	26.91733086	11.69436534	3.151963385
70	27.79306894	11.56843528	2.32947738
80	26.26756105	13.42580369	3.836421126
90	28.23834577	13.79877455	3.38180122
100	29.58797139	13.41935483	3.981139163

Table 4.9: The Average delay of the multicasted messages

Number of Islands	Multicasting Technology		
	ALM	ONM	NM
10	232338115.2	2943908.9	122092
25	135570951.2	3089931.5	48343
50	84229704.2	3380920.3	24125
75	64081765.9	3721394.6	16054
100	52619581.7	4095289.6	12177
150	42560863.6	5142930.1	8009

Table 4.10: Average Number of messages in the island

Number of Islands	Multicasting Technology		
	ALM	ONM	NM
10	726.5583171	1	10.79934783
25	1066.396257	1	25.89858696
50	1116.967909	1	51.29413043
75	1111.108614	1	76.39858696
100	1052.658583	1	101.1879348
150	1006.515662	1	149.3146739

Table 4.11: Average stress on the backbone

Number of Islands	Multicasting Technology		
	ALM	ONM	NM
1	8.022049128	2.460548928	1
10	7.905024619	3.417358352	1
20	7.980704276	3.961982012	1
30	8.141042514	4.277412383	1
40	8.139815375	4.449511377	1
50	8.128674458	4.598524242	1
60	8.07945546	4.710337911	1
70	8.079224642	4.819346505	1
80	8.045887372	4.988823953	1
90	8.136124328	5.002588781	1
100	8.0591473	4.925337349	1

Table 4.12: Average stretch of the multicasted message



## CONCLUSION AND FUTURE WORK

---

### 5.1 INTRODUCTION

Currently, the Internet is fragmented into many multicast islands due to the lack of support of native multicast in the backbone and because different types/protocols for native multicast are deployed. This is a major issue for content distribution as multicast is not usable if nodes in multiple islands are involved in a transmission.

In this paper, we have addressed this problem by proposing Opportunistic Native Multicast (ONM) which is a novel approach that joins together islands of native multicast capable region of the Internet. ONM uses a P2P overlay network to discover and connect islands. In doing so, it uses Automatic Multicast Tunnelling (AMT), which was standardised in IETF RFC 7450, to forward data between these islands. ONM combines network resources, employs native multicast where possible and falls back to Application Layer operations where needed.

The proposed approach Opportunistic Native Multicast (ONM) was compared against of the existing solutions, Island Multicast (IM). The thesis compared the two approaches and identified multiple areas where ONM is better. Also, we compared the performance of ONM with the IM where ONM provided better success rate and lower control overhead.

We tested our approach under a number of different realistic network configurations and churn settings, together with different algorithms to select primary and secondary nodes in the islands. Our experimentation suggest that ONM should be implemented with a Multi-Hop Overlay which supports an Application Layer Multicast approach such as Pastry and Scribe. We further recommend the use of node age as the factor to select primary and secondary nodes. Results suggest that the use of a single secondary node is desirable and additional backup nodes

do not yield a significant performance gain. According to our testing the multiplier value  $N$  should be set to a value of 4. Higher values do not result in better performance.

## 5.2 CONTRIBUTIONS

This thesis was able to produce multiple significant contributions that are discussed in this section. These contributions were the objectives of our research and were discussed in Chapter 1.

### 5.2.1 *Review of current literature*

For the gap to be identified, we reviewed related technical subjects in Chapter 2. The chapter reviewed the issue of multicast islands and the related technologies: Peer To Peer (P2P), Multicasting and Automatic Multicast Tunnelling (AMT). Moreover, the existing proposed hybrid multicast techniques were analysed and discussed and shortcomings identified.

### 5.2.2 *Connecting Multicast Islands using AMT and ALM*

The proposed framework introduces the use of Automatic Multicast Tunnelling (AMT) built using an Application Layer Multicast (ALM) overlay to connect islands and peers in a unicast only network. The resulting tree of AMT tunnels is maintained by the protocol using the Peer To Peer (P2P) overlay in the ALM. While some approaches were introduced to solve the issue of Multicast Islands, they can be improved by introducing AMT tunnelling. However, AMT on its own does not offer the needed capability to let multiple AMT nodes negotiate and elect one of them to act as Automatic Multicast Tunnelling (AMT) gateways and relays. The election algorithms and the integration of AMT with ALM allow for the automatic discovery of islands and the selection of nodes as gateways and

relays. The proposed Opportunistic Native Multicast (ONM) has one elected node at each island that acts as a Primary Node (PN). This node is part of the ALM tree and it also terminates the AMT tunnel. The Primary Node (PN) is responsible for bridging the local Native Multicast (NM) tree with the inter-islands overlay tree. Also, peers that are in an unicast only network are able to connect to the multicast tree by considering them self an island of one.

### 5.2.3 *Detect Nodes Failure*

The thesis propose a process for a failure to be detected. The ONM uses frequent and periodic hello messages to allow other nodes to declare other node to be dead when multiple subsequent hello messages was detected.

The number of missed heartbeats needed to declare a node dead  $N$  is important parameters as it controls the trade-off between faster detection of dead node and falsely declaring a node dead and taking over its responsibility by another node. So, by reacting faster to failures, the island minimises the down time of waiting for the another node to resumes the responsibility of the failed node. Alternatively, falsely declaring a node dead unnecessarily makes a backup node takes over or, in the worst case, trigger a reelection. During the election period, the island is considered down until another node takes over. According to the simulation results, a value of 4 is good balance for the value of  $N$ .

### 5.2.4 *Availability and resilience*

Since the Primary Node (PN) is susceptible to failure and churn, the thesis discussed efficient mechanisms of how a failure is detected in the island and for the process to elect the new PN to take place. Also, to allow for faster recovery time after Primary Node (PN) failure, the concept of Secondary Node (SN) was introduced. SN acts as a backup node that helps in localising the control traffic as shown in Chapter 4. Also, the thesis discussed the use of multiple SNs. It was

found that while using a Secondary Node (SN) improves the stability and the success rate, using additional SN do not noticeably improve the results.

#### 5.2.5 *Distinguishing between low and high churn nodes in a heterogeneous network*

Realistically, nodes in the islands have different churn rate and stability probabilities. So, it is better for the stability of the network to select low churn nodes to act as PN or SN. The thesis suggests the use and taking advantage of the age of the node to determine the likelihood that the node has a stable churn to be suitable to act as Primary or Secondary Nodes. To determine the improvements achieved by factoring the age of the node, it was compared against two other approaches: Manual and Random. It was found that factoring age improves performance over randomly selecting nodes.

#### 5.2.6 *Dynamically Optimising control traffic*

Since the stability of the newly elected Primary Node (PN) can vary across the lifetime of the network as nodes get elected to be PN or SN. This helps in reducing the control traffic produced in the island without sacrificing the fast recovery time of the islands in the case of high churn networks. So, the thesis discussed two approaches for an island to dynamically tune down the frequency of control messages when a stable network is detected: Probation Time and Gradual Trust. It was found that dynamical optimising the control traffic reduces the overhead without affecting the success rate. Also, using Graduate Trust has a better performance in term of overhead.

#### 5.2.7 *Performance of ONM*

Island Multicast (IM) is one the main implementations of Hybrid Multicast (HM) as discussed in Section 2.1.4.2. It was chosen as a benchmark for ONM to be compared

to. In Section 3.1.3, the main advantages offered by ONM was reviewed. Moreover, in Section 4.10, the performance of these two approaches were compared. It can be noticed that ONM offers better performance and many more advantages than IM

#### 5.2.8 *The simulator*

First of all, the thesis investigated the use of the current testing frameworks and simulators environments. Due to the lack of support for Hybrid Multicast (HM) and Automatic Multicast Tunnelling (AMT), a simulator was modified to be able to accomplish these tasks. The simulator module was integrated with multiple statistics collections signals that are used to measure and compare the performance of our approach. Due to its modularity and extensibility, OMNeT++ was selected as the simulator.

### 5.3 LIMITATIONS AND FUTURE WORK

While this theses was able to achieve the contributions listed in Section 5.2, some limitations were faced in the research. These limitations are identified in this section and the potential future work are reviewed.

The theses identified the following limitations:

**REAL WORLD DEPLOYMENT** This thesis has implemented and tested the proposed framework using a simulator due to reasons discussed in Section 4.2. While the individual parts that comprise ONM are already implemented (See FreePastry [11] for Pastry, JANUS [12] which implements SCRIBE, AMT is implemented in some routers [98]), they have not been integrated into a single operational system to date. A real world implementation of Opportunistic Native Multicast (ONM) will provide better evaluation of more aspects than the simulation. Some factors, such as: network topology, traffic pattern and user behaviour, can affect the results in a way different than

the simulation can capture [94]. Other approaches of Hybrid Multicast (HM) used testbeds and simulations instead of using a real world implementation, such as Island Multicast (IM) [99] and [100].

**SECURITY** As discussed in Section 2.1.4.5, security is identified as one the main issues facing deploying Hybrid Multicast. While Security is too big of a subject to be include as part of this thesis, and duo its importance, it is beyond the scope of this thesis. Nonetheless, two main components that are part of the implementation of Opportunistic Native Multicast (ONM) can improve security:

1. The security of Pastry can be improved by Security Enhanced Pastry (SEPastry) [101].
2. AMT can use hashes and Message Authentication Code (MAC) to secure the tunnels.

Besides these points, the overall system needs to be looked at separately.

**DIFFERENT OVERLAY** While Pastry was recommended and choose for this implementation of Opportunistic Native Multicast (ONM), other overlays can be used instead. Moreover, different types of Peer To Peer (P2P) systems such as a single hop systems can be used as they have faster time is locating resources in the overlay. However, multihop overlays were chosen for their scalability.

**CONSIDER OTHER FACTORS FOR SELECTING PRIMARY NODES** This thesis has considered multiple mechanisms for selecting a Primary Node (PN). However, it will be worthwhile to consider other factors such as the location of the node in the islands and the available resources.

**MULTIPLE NATIVE MULTICAST PROTOCOLS** The design of Opportunistic Native Multicast (ONM) does not impose any restriction on the type of the Native Multicast (NM) protocol. So, different approaches can be used. While this was not tested at this stage, it is very helpful to implement it and study

its effects as it is one of the main reasons multicast islands exists in the first place.

**MOBILITY** In real life, and with the wide spread of mobile nodes, mobility is an issue that should be given more emphases when designing new systems. However, Peer To Peer (P2P) can handle the movement of nodes which could be used to improve detecting and handling the mobility of nodes [102].

#### 5.4 SUMMARY

This chapter highlights the contributions of this thesis. Firstly, the achieved objectives of the thesis were listed. Secondly, based on the experimental results presented in Chapter 4, some of the major findings and recommendations were discussed. After simulating Opportunistic Native Multicast (ONM) under different configurations, it is possible to draw the following quantitative conclusions:

- ONM has lower stretch compared to ALM. Figure 4.10 shows a configuration where ONM's stretch was less than half of ALM's.
- Using ONM has reduced the stress on the backbone. In some cases, such as in Figure 4.11, the stress was around 10 and 1000 for ONM and ALM respectively.
- Using ONM has reduced the number of packets crossing the backbone exponentially. When compare to Application Layer Multicast (ALM) as in Figure 4.13, reduction by two orders of magnitude can be achieved.
- Using ONM has lower delay in distributing the Multicast messages. For example, the delay was reduced by more 50% as seen in Figure 4.14
- It was found that while using a Secondary Node (SN) improves the stability and the success rate, using additional SN do not noticeably improve the results. So, a single SN is enough.

- Increasing the frequency of heartbeats improves the operation of ONM to a point. After that, it will increase the control traffic without much improvements in operation.
- The number of missed heartbeats needed to declare a node dead  $N$  is important parameters as it controls the trade-off between faster detection of dead node and falsely declaring a node dead and taking over its responsibility by another node. It was found in the simulation, see Figure 4.28, that using a value of 4 provides a good balance.
- Factoring the age of the node in selecting Primary and Secondary Nodes will improve the results as seen in Figure 4.25.
- It was found that dynamical optimising the control traffic reduces the overhead without affecting the success rate.
- Also, using Graduate Trust has a better performance in term of overhead.
- Compared to Island Multicast (IM), Opportunistic Native Multicast (ONM) offers better performance and many more advantages.

Finally, the limitations facing the body of work reported in this thesis were listed. These limitations serve as a base for the future work that should be further investigated.



## BIBLIOGRAPHY

---

- [1] M. Kolberg, "Employing Multicast in P2P Overlay Networks," pp. 1–17.
- [2] M. Kolberg and J. Buford, "Application Layer Multicast extensions to RELOAD," *2011 IEEE Consumer Communications and Networking Conference (CCNC)*, pp. 1083–1087, 1 2011. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epico3/wrapper.htm?arnumber=5766334>
- [3] CISCO, "Cisco Visual Networking Index : Forecast and Methodology , 2012 - 2017," *White Paper*, pp. 1–10, 2013.
- [4] D. Alwadani, M. Kolberg, and J. Buford, "A Simulation Model for Hybrid Multicast," *2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies*, pp. 112–116, 2014. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epico3/wrapper.htm?arnumber=6982901>
- [5] M. Kobayashi, H. Nakayama, N. Ansari, and N. Kato, "Robust and efficient stream delivery for application layer multicasting in heterogeneous networks," *IEEE Transactions on Multimedia*, vol. 11, no. 1, pp. 166–176, 2009.
- [6] C. Diot, B. Levine, and B. Lyles, "Deployment issues for the IP multicast service and architecture," *Network, . . .*, pp. 1–18, 2000. [Online]. Available: [http://ieeexplore.ieee.org/xpls/abs\\_all.jsp?arnumber=819174](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=819174)
- [7] R. Boivie, N. Feldman, Y. Imai, W. Livens, and D. Ooms, "Explicit multicast (Xcast) concepts and options," Tech. Rep., 2007.
- [8] X. Jin, T. Ho-Shing, S. Chan, and K.-L. Cheng, "Deployment issues in scalable island multicast for peer-to-peer streaming," *IEEE multimedia*, pp. 72–80, 2009. [Online]. Available: <http://repository.ust.hk/dspace/handle/1783.1/6581>

- [9] H. ERIKSSON, "MBone: the multicast backbone," *Communications of the ACM*, vol. 37, pp. 54–60, 1994.
- [10] M. Wählisch, S. Venaas, and T. Schmidt, "A common API for transparent hybrid multicast," pp. 1–42, 2012. [Online]. Available: <http://tools.ietf.org/html/irtf-samrg-common-api-07.txt>
- [11] P. Druschel, E. Engineer, R. Gil, Y. C. Hu, S. Iyer, A. Ladd, and others, "FreePastry," *Software available at http://www.cs.rice.edu/CS/Systems/Pastry/-FreePastry*, 2001.
- [12] R. Riggio, N. Scalabrino, D. Miorandi, and I. Chlamtac, "Janus: A framework for distributed management of wireless mesh networks," in *Testbeds and Research Infrastructure for the Development of Networks and Communities*, 2007. *TridentCom 2007. 3rd International Conference on*. IEEE, 2007, pp. 1–7.
- [13] C. Westphal, "Challenges in Networking to Support Augmented Reality and Virtual Reality," *ICNC 2017*, 2017.
- [14] A. Ali, J. Qadir, A. Sathiaseelan, K.-L. A. Yau, and J. Crowcroft, "MP-ALM: Exploring Reliable Multipath Multicast Streaming with Multipath TCP," in *Local Computer Networks (LCN), 2016 IEEE 41st Conference on*, 2016, pp. 138–146.
- [15] J. Huang, Q. Duan, Y. Zhao, Z. Zheng, and W. Wang, "Multicast Routing for Multimedia Communications in the Internet of Things," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 215–224, 2 2017.
- [16] T. Ruso, C. Chellappan, and P. Sivasankar, "Ppssm:push/pull smooth video streaming multicast protocol design and implementation for an overlay network," *Multimedia Tools and Applications*, vol. 75, no. 24, pp. 17 097–17 119, 12 2016. [Online]. Available: <https://doi.org/10.1007/s11042-015-2979-5>
- [17] V. Rabarijaona, F. Kojima, H. Harada, and C. Powell, "Enabling Layer 2 Routing in IEEE std 802.15.4 Networks with IEEE std 802.15.10," *IEEE Communications Standards Magazine*, vol. 1, no. 1, pp. 44–49, 3 2017.

- [18] T. A. Le, H. Nguyen, and M. C. Nguyen, "Application-network cross layer multi-variable cost function for application layer multicast of multimedia delivery over convergent networks," *Wireless Networks*, vol. 21, no. 8, pp. 2677–2692, 11 2015. [Online]. Available: <https://doi.org/10.1007/s11276-015-0940-1>
- [19] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, *Scalable application layer multicast*. ACM, 2002, vol. 32, no. 4.
- [20] J.-M. Vella and S. Zammit, "A survey of multicasting over wireless access networks," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp. 718–753, 2013.
- [21] P. Paul and S. V. Raghavan, "Survey of multicast routing algorithms and protocols," in *Proceedings of the international conference on computer communication*, vol. 15, no. 3, 2002, p. 902.
- [22] L. Wei and D. Estrin, "A comparison of multicast trees and algorithms," *Submitted to INFOCOM*, vol. 94, 1993.
- [23] J. Moy, "Multicast routing extensions for OSPF," *Communications of the ACM*, vol. 37, no. 8, pp. 61–67, 1994.
- [24] S. Bhattacharyya, "An overview of source-specific multicast (SSM)," 2003.
- [25] N. Mir, S. Musa, R. Torresand, and S. Swamy, "Evaluation of PIM and CBT multicast protocols on fault-tolerance," *International Journal of Computing and Networking Technology*, vol. 2, no. 2, pp. 59–64, 2014.
- [26] M. Castro and P. Druschel, "SCRIBE: A large-scale and decentralized application-level multicast infrastructure," *Selected Areas in ...*, vol. 20, no. 8, 2002. [Online]. Available: [http://ieeexplore.ieee.org/xpls/abs\\_all.jsp?arnumber=1038579](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1038579)
- [27] A. Adams, J. Nicholas, and W. Siadak, "Protocol independent multicast-dense mode (PIM-DM): Protocol specification (revised)," Tech. Rep., 2004.

- [28] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos, "Multicast routing for multimedia communication," *IEEE/ACM Transactions on Networking (TON)*, vol. 1, no. 3, pp. 286–292, 1993.
- [29] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for Steiner trees," *Acta informatica*, vol. 15, no. 2, pp. 141–145, 1981.
- [30] B. Waxman, "Delay-bounded Steiner Tree Algorithm for Performance-Driven Layout," *Journal on Selected Areas in Communications*, vol. 6, pp. 1617–1622, 1988.
- [31] J.-J. Pansiot and D. Grad, "On routes and multicast trees in the Internet," *ACM SIGCOMM Computer Communication Review*, vol. 28, no. 1, pp. 41–50, 1998.
- [32] Y. H. Tsin, "Incremental distributed asynchronous algorithm for minimum spanning trees," *Computer networks and ISDN systems*, vol. 26, no. 2, pp. 227–232, 1993.
- [33] Q. Zhu, M. Parsa, and J. J. Garcia-Luna-Aceves, "A source-based algorithm for delay-constrained minimum-cost multicasting," in *INFOCOM'95. Fourteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Bringing Information to People. Proceedings. IEEE*, vol. 1, 1995, pp. 377–385.
- [34] F. Bauer and A. Varma, "Distributed algorithms for multicast path setup in data networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 4, no. 2, pp. 181–191, 1996.
- [35] G. N. Rouskas and I. Baldine, "Multicast routing with end-to-end delay and delay variation constraints," *IEEE Journal on Selected Areas in communications*, vol. 15, no. 3, pp. 346–356, 1997.
- [36] F. Bauer and A. Varma, "ARIES: A rearrangeable inexpensive edge-based on-line Steiner algorithm," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 3, pp. 382–397, 1997.

- [37] J. Westbrook and D. C. K. Yan, "Greedy algorithms for the on-line Steiner tree and generalized Steiner problems," in *Workshop on Algorithms and Data Structures*, 1993, pp. 622–633.
- [38] T. Alrabiah and T. F. Znati, "Delay-bounded Online Multicasting," in *High Performance Networking: IFIP TC-6 Eighth International Conference on High Performance Networking (HPN '98) Vienna, Austria, September 21–25, 1998*, vol. 8, 2013, p. 95.
- [39] L. Aguilar, "Datagram routing for internet multicasting," in *ACM SIGCOMM Computer Communication Review*, vol. 14, no. 2. ACM, 1984, pp. 58–63.
- [40] M. Brogle, D. Milic, L. Bettosini, and T. Braun, "A performance comparison of native IP Multicast and IP Multicast tunneled through a Peer-to-Peer overlay network," *2009 International Conference on Ultra Modern Telecommunications & Workshops*, pp. 1–6, 10 2009. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epico3/wrapper.htm?arnumber=5345658>
- [41] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A scalable content-addressable network," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 161–172, 10 2001. [Online]. Available: <http://portal.acm.org/citation.cfm?doid=964723.383072>
- [42] S. Ratnasamy and M. Handley, "Application-level multicast using content-addressable networks," *Networked Group ...*, pp. 14–29, 2001. [Online]. Available: [http://link.springer.com/chapter/10.1007/3-540-45546-9\\_2](http://link.springer.com/chapter/10.1007/3-540-45546-9_2)
- [43] B. Zhang, S. Jamin, and L. Zhang, "Universal IP multicast delivery," *International Workshop on Networked Group Communication*, 2002.
- [44] G. Bumgardner, "Automatic Multicast Tunneling," pp. 1–82, 2 2015. [Online]. Available: <https://www.rfc-editor.org/info/rfc7450>

- [45] A. Sethi, S. Suthar, V. Yadav, and A. Kumar, "A Survey of QoS Multicast Protocols for MANETs," *Journal of Network Communications and Emerging Technologies (JNCET) www.jncet.org*, vol. 6, no. 3, 2016.
- [46] Y. Gu, L. Breslau, N. Duffield, and S. Sen, "GRE Encapsulated Multicast Probing: A Scalable Technique for Measuring One-Way Loss," in *2008 IEEE INFOCOM - The 27th Conference on Computer Communications*. IEEE, 4 2008, pp. 1651–1659. [Online]. Available: [http://ieeexplore.ieee.org/xpls/abs\\_all.jsp?arnumber=4509821](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4509821)<http://ieeexplore.ieee.org/lpdocs/epico3/wrapper.htm?arnumber=4509821>
- [47] X.-j. Liu, H.-r. Zhong, C.-q. Yie, and S.-y. Jin, "A Hybrid IP Multicast Based Data Dissemination Allocation Strategy in Large Scale Distributed Simulations," *Acta Electronica Sinica*, vol. 31, no. 11, pp. 1678–1681, 2003.
- [48] K. Tan, Y.-c. SHI, C.-y. LIAO, and G.-y. XU, "An Application-level Semantic Reliable Multicast Architecture For the Internet," *Journal of Software*, vol. 4, p. 11, 2002.
- [49] B. Zhang, W. Wang, S. Jamin, D. Massey, and L. Zhang, "Universal IP multicast delivery," *Computer Networks*, vol. 50, no. 6, pp. 781–806, 4 2006. [Online]. Available: <http://linkinghub.elsevier.com/retrieve/pii/S1389128605002410>
- [50] X. Jin, K. Cheng, and S. Chan, "Island multicast: combining IP multicast with overlay data distribution," *Multimedia, IEEE Transactions ...*, vol. 11, no. 5, pp. 1024–1036, 2009. [Online]. Available: [http://ieeexplore.ieee.org/xpls/abs\\_all.jsp?arnumber=4907113](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4907113)
- [51] S. Jamin, L. Zhang, and B. Zhang, "Host multicast: a framework for delivering multicast to end users," *Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, pp. 1366–1375, 2002. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epico3/wrapper.htm?arnumber=1019387>

- [52] S. Cho and M.-S. Park, "FJM: fast join mechanism for overlay multicast," in *Control Applications, 2003. CCA 2003. Proceedings of 2003 IEEE Conference on*, vol. 2. IEEE, 2003, pp. 1333–1338.
- [53] J. Park, D. Y. Kim, S. G. Kang, and S. J. Koh, "Multicast delivery based on unicast and subnet multicast," *IEEE Communications Letters*, vol. 5, no. 4, pp. 181–183, 4 2001. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epico3/wrapper.htm?arnumber=917107>
- [54] A. Wacker, G. Schiele, S. Holzapfel, and T. Weis, "A NAT Traversal Mechanism for Peer-To-Peer Networks," in *2008 Eighth International Conference on Peer-to-Peer Computing*. IEEE, 9 2008, pp. 81–83. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epico3/wrapper.htm?arnumber=4627263>
- [55] Y. Wang, Z. Lu, and J. Gu, "Research on Symmetric NAT Traversal in P2P applications," in *2006 International Multi-Conference on Computing in the Global Information Technology - (ICCGI'06)*. IEEE, 2006, pp. 59–59. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epico3/wrapper.htm?arnumber=4124078>
- [56] M. Cinque, D. Cotroneo, and M. Fiorentino, "Facing reliability requirements for timely information sharing in future crisis management systems," in *Fast Abstract in the 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2016.
- [57] M. Cai, X. Wen, W. Zheng, Y. Cheng, and Y. Sun, "Different-Strategy Management of Malicious Nodes in the Peer-to-Peer Network," in *2009 International Conference on Environmental Science and Information Application Technology*. IEEE, 7 2009, pp. 575–578. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epico3/wrapper.htm?arnumber=5199759>
- [58] Y. I. Loubna Mekouar, "Detecting malicious peers in a reputation-based peer-to-peer system," in *Second IEEE Consumer Communications*

- and Networking Conference, 2005. CCNC. 2005. IEEE, pp. 37–42. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epico3/wrapper.htm?arnumber=1405140>*
- [59] Q. H. Vu, M. Lupu, and B. C. Ooi, *Peer-to-peer computing: Principles and applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. [Online]. Available: <http://www.springerlink.com/index/10.1007/978-3-642-03514-2>
- [60] J. F. Buford and H. Yu, “Peer-to-Peer Networking and Applications: Synopsis and Research Directions,” in *Handbook of Peer-to-Peer Networking*. Springer US, 2010, pp. 3–45.
- [61] J. F. Buford, H. Yu, and E. K. Lua, *P2P Networking and Applications*. Burlington, USA: Morgan Kaufmann, Elsevier, 2009.
- [62] “OpenNap: Open Source Napster Server,” 2001. [Online]. Available: <http://opennap.sourceforge.net/>
- [63] T. Klingberg and R. Manfredi, “Gnutella 0.6,” 2002. [Online]. Available: [http://rfc-gnutella.sourceforge.net/src/rfc-o\\_6-draft.html](http://rfc-gnutella.sourceforge.net/src/rfc-o_6-draft.html)
- [64] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, “Making gnutella-like p2p systems scalable,” in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM, 2003, pp. 407–418.
- [65] M. Duigou, “JXTA Protocol,” 2002. [Online]. Available: <http://tools.ietf.org/pdf/draft-duigou-jxta-protocols-00>
- [66] M. Castro, M. Costa, and A. Rowstron, “Peer-to-peer overlays: structured, unstructured, or both?” *Technical Report*, 2004. [Online]. Available: <http://research.microsoft.com/en-us/um/people/antr/ms/structella-tr.pdf>
- [67] D. Korzun and A. Gurtov, *Structured Peer-to-Peer Systems : Fundamentals of Hierarchical Organization, Routing, Scaling, and Security*. New York: Springer Science + Business Media, 2013.



- [68] K. Dhara, Y. Guo, M. Kolberg, and X. Wu, "Overview of Structured Peer-to-Peer Overlay Algorithms," in *Handbook of Peer-to-Peer Networking*. US: Springer, 2010, pp. 223–256.
- [69] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing*. Springer, 2001, pp. 329–350.
- [70] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 149–160, 2001.
- [71] P. Maymounkov and D. Mazieres, "Kademlia: A Peer-to-peer Information System Based on the {XOR} Metric," in *Proc. 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, F. Kaashoek and A. Rowstron, Eds., 3 2002.
- [72] C. G. Plaxton, R. Rajaraman, and A. W. Richa, "Accessing nearby copies of replicated objects in a distributed environment," *Theory of Computing Systems*, vol. 32, no. 3, pp. 241–280, 1999.
- [73] M. Castro, P. Druschel, A.-M. Kermarrec, and A. I. T. Rowstron, "SCRIBE: A large-scale and decentralized application-level multicast infrastructure," *IEEE Journal on Selected Areas in communications*, vol. 20, no. 8, pp. 1489–1499, 2002.
- [74] B. Leong, B. Liskov, and E. Demaine, "EpiChord: parallelizing the chord lookup algorithm with reactive routing state management," in *Proceedings. 2004 12th IEEE International Conference on Networks (ICON 2004) (IEEE Cat. No.04EX955)*. IEEE, pp. 270–276. [Online]. Available: <http://ieeexplore.ieee.org/document/1409145/>
- [75] A. Gupta, B. Liskov, R. Rodrigues, and others, "One Hop Lookups for Peer-to-Peer Overlays." in *HotOS*, 2003, pp. 7–12.

- [76] L. R. Monnerat and C. L. Amorim, "D<sub>1</sub>HT : A Distributed One Hop Hash Table," in *International Parallel and Distributed Processing Symposium (IPDPS)*, Rhodes Island, 2006.
- [77] F. Chowdhury and M. Kolberg, "Performance evaluation of structured Peer-to-Peer Overlays for Use on Mobile Networks," in *6th International conference on Developments in eSystems Engineering (DESE)*. Abu Dhabi: IEEE, 2013, pp. 57–62.
- [78] J. Furness, F. Chowdhury, and M. Kolberg, "An Evaluation of EpiChord in OverSim," in *the Fifth International Conference on Networks & Communications (NetCom)*, ser. Lecture Notes in Electrical Engineering, vol. 284. Chennai, India: Springer International Publishing, 2014, pp. 3–19.
- [79] F. Chowdhury and M. Kolberg, "Performance Evaluation of EpiChord under High Churn," in *the 8th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks (PM2HW2N)*, Barcelona, Spain, 2013, pp. 29–36.
- [80] J. Li, J. Stribling, R. Morris, and M. F. Kaashoek, "Bandwidth-efficient management of DHT routing tables," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*. USENIX Association, 2005, pp. 99–114.
- [81] A. Brown, M. Kolberg, and J. F. Buford, "Chameleon: an adaptable 2-tier variable hop overlay," in *2009 6th IEEE Consumer Communications and Networking Conference*. IEEE, 2009, pp. 1–6.
- [82] I. Clarke, O. Sandberg, B. Wiley, and T. Hong, "Freenet: A distributed anonymous information storage and retrieval system," in *Designing Privacy Enhancing Technologies*. Berlin Heidelberg: Springer, 2001, pp. 46–66.
- [83] "The Annotated Gnutella Protocol Specification vo.4." [Online]. Available: <http://rfc-gnutella.sourceforge.net/developer/stable/>

- [84] S. Androutsellis-Theotokis and D. Spinellis, "A survey of peer-to-peer content distribution technologies," *ACM Computing Surveys*, vol. 36, no. 4, pp. 335–371, 12 2004. [Online]. Available: <http://portal.acm.org/citation.cfm?doid=1041680.1041681>
- [85] G. Bumgardner, "Automatic Multicast Tunneling," pp. 1–86, 2012. [Online]. Available: <http://tools.ietf.org/html/ietf-mboned-auto-multicast-14>
- [86] T. Kernén and S. Simlo, "AMT - Automatic IP Multicast without explicit Tunnels," pp. 1–11, 2010.
- [87] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Application-level multicast using content-addressable networks," in *International Workshop on Networked Group Communication*. Springer, 2001, pp. 14–29.
- [88] I. Baumgart, B. Heep, and S. Krause, "OverSim: A Flexible Overlay Network Simulation Framework," *2007 IEEE Global Internet Symposium*, pp. 79–84, 5 2007. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4301435>
- [89] J. Furness, M. Kolberg, and M. Fayed, "An evaluation of Chord and Pastry models in OverSim," in *Modelling Symposium (EMS), 2013 European*. IEEE, 2013, pp. 509–513.
- [90] Z. Yao, X. Wang, D. Leonard, and D. Loguinov, "Node isolation model and age-based neighbor selection in unstructured P2P networks," *IEEE/ACM Transactions On Networking*, vol. 17, no. 1, pp. 144–157, 2009.
- [91] N. Kotilainen, M. Vapa, T. Keltanen, A. Auvinen, and J. Vuori, "P2PRealm - Peer-to-Peer Network Simulator," in *2006 11th International Workshop on Computer-Aided Modeling, Analysis and Design of Communication Links and Networks*, vol. 35. IEEE, 2006, pp. 93–99. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1649724>

- [92] J. F. Buford and M. Kolberg, "Hybrid overlay multicast simulation and evaluation," in *Consumer Communications and Networking Conference, 2009. CCNC 2009. 6th IEEE*, 2009, pp. 1–2.
- [93] D. Stutzbach and R. Rejaie, "Understanding churn in peer-to-peer networks," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, 2006, pp. 189–202.
- [94] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspar, "How physical network topologies affect virtual network embedding quality: A characterization study based on ISP and datacenter networks," *Journal of Network and Computer Applications*, vol. 70, pp. 1–16, 2016.
- [95] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Middleware*. Berlin Heidelberg: Springer, 2001, vol. 2218, pp. 329–350.
- [96] A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment," ... of the 1st international conference on Simulation ..., 2008. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1416290>
- [97] D. Alwadani, M. Kolberg, and J. Buford, "An evaluation of Opportunistic Native Multicast," in *Computer Aided Modelling and Design of Communication Links and Networks (CAMAD), 2015 IEEE 20th International Workshop on*. IEEE, 2015, pp. 170–174.
- [98] A. Headquarters, "Cisco IOS XR MPLS Configuration Guide for the Cisco CRS-1 Router, Release 3.9," 2009.
- [99] K.-L. Cheng, K.-W. Cheuk, and S.-H. Chan, "Implementation and performance measurement of an island multicast protocol," in *Communications, 2005. ICC 2005. 2005 IEEE International Conference on*, vol. 2, 2005, pp. 1299–1303.
- [100] M. Won and R. Stoleru, "A Hybrid Multicast Routing for Large Scale Sensor Networks with Holes," *IEEE Transactions on Computers*, vol. 64, no. 12, pp. 3362–3375, 12 2015.

- [101] M. Mishra, S. Tripathy, and S. Peri, "SEPastry: Security Enhanced Pastry," in *Advances in Computing and Information Technology: Proceedings of the Second International Conference on Advances in Computing and Information Technology (ACITY) July 13-15, 2012, Chennai, India - Volume 1*, N. Meghanathan, D. Nagamalai, and N. Chaki, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 789–795. [Online]. Available: [http://dx.doi.org/10.1007/978-3-642-31513-8\\_80](http://dx.doi.org/10.1007/978-3-642-31513-8_80)
- [102] L. Li, A. Gaddah, and T. Kunz, "Mobility support in a tactical P2P publish/subscribe overlay," in *Military Communications Conference, 2008. MILCOM 2008. IEEE*, 2008, pp. 1–7.