TR2005 25 S

Stellingen
behorende bij het proefschrift
'Multicast in Network and Application Layer'
Milena Janić

1. Hoewel de *random graph* geen goed model is voor de topologie van het Internet, is de kortste-pad-boom afgeleid van die *graph* een redelijke benadering voor de structuur van multicastbomen in het Internet. (dit proefschrift)
2. Indien de onderliggende topologie van het Internet volledig onbekend is voor de eindgebruikers, is de implementatie van applicatielaag multicast zinloos. (dit proefschrift)
3. Men moet voorzichtig zijn met het trekken van conclusies op basis van een onvolmaakt instrument als *traceroute.* (dit proefschrift)
4. Het model voor multicast bomen voorgesteld in dit proefschrift, en de berekeningen op basis daarvan, duiden aan dat netwerklaag multicast, voor de redelijk breedbandige applicaties, al bij een beperkte aantal gebruikers voor ISPs voordeliger dan unicast wordt. (dit proefschrift)
5. Zolang er geen "killer applicatie" voor netwerklaag multicast bestaat, wordt netwerklaag multicast niet door ISPs ingevoerd. En zolang die niet wordt ingevoerd is er geen bodem voor het onstaan van zo'n applicatie. Dit kip-en-ei probleem van netwerklaag multicast wordt met applicatielaag multicast opgelost.
6. Het probleem met terrorisme is dat zelfs mislukte pogingen zijn gelukt.
7. Onverschilligheid moet niet met tolerantie worden verward.
8. 'Humanitaire interventies' kunnen het verantwoordelijkheidsgevoel aflossen, maar kunnen meer schade dan goed aan de lijdende burgers brengen.
9. Het zou de democratie ten goede kunnen komen als een stem gewogen kon worden met de kennis van de kiezer van waarop hij stemt.
10. Elektrotechnisch onderzoek heeft in Nederland dezelfde aantrekkingskracht als schoonmaakwerk. Het wordt vooral door buitenlanders verricht.

Deze stellingen worden opponeerbaar en verdedigbaar geacht en zijn als zodanig goedgekeurd door de promotor,

Prof.dr.ir. P. F.A. Van Mieghem

Propositions
accompanying the thesis
'Multicast in Network and Application Layer'
Milena Janić

1. Although the random graph is not a good model for the topology of the Internet, the shortest path tree derived from such graph seems to be a reasonable approximation for the structure of multicast trees in the Internet. (this thesis)
2. If the underlying topology is unknown to the end-users, the implementation of application layer multicast is useless. (this thesis)
3. One must be extremely careful when drawing conclusions based on such an imperfect utility as *traceroute.* (this thesis)
4. The model for multicast we propose, and the calculations based on it, indicate that for reasonably bandwidth-demanding applications, already for a limited number of users multicast becomes more beneficial than unicast for IPSs. (this thesis)
5. As long as there is no "killer application" for network layer multicast, network layer multicast will not be deployed by the ISPs. And as long as it is not deployed, there will be no basis to develop such an application. This chicken-and-egg problem of network layer multicast will be solved by application layer multicast.
6. The problem with terrorism is that even failed attempts are successful.
7. Indifference must not be mistaken for tolerance.
8. 'Humanitarian interventions' can relieve the sense of responsibility, but could bring more harm than good to the suffering nations.
9. The democracy could benefit if the votes could be weighted by the knowledge of the voter about what is being voted on.
10. The research in the field of electrical engineering has the same attraction as cleaning work. In the Netherlands they are both being performed mainly by foreigners.

These propositions are considered opposable and defendable and as such have been approved by the supervisor,

Prof.dr.ir. P. F.A. Van Mieghem

TR200525

Multicast in Network and Application Layer

Multicast in Network and Application Layer

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof.dr.ir. J.T. Fokkema,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op maandag 31 oktober 2005 om 15:30 uur

door

Milena JANIĆ

Electrical Engineer van de Universiteit van Belgrado, Servië en Montenegro,
geboren te Belgrado, Servië en Montenegro.

Dit proefschrift is goedgekeurd door de promotor:
Prof.dr.ir. P.F.A. Van Mieghem

Samenstelling promotiecommissie:

Rector Magnificus,	Voorzitter
Prof.dr.ir. P.F.A. Van Mieghem,	Technische Universiteit Delft, promotor
Prof.dr.ir. I.G.M.M. Niemegeers,	Technische Universiteit Delft
Prof.dr.ir. S. Vassiliadis,	Technische Universiteit Delft
Prof.dr.ir. H. Sips,	Technische Universiteit Delft
Prof.dr.ir. M. van Steen,	Vrije Universiteit Amsterdam
Prof.dr. F.P.L. Boavida,	University of Coimbra, Portugal
Prof.dr.ir. N.H.G. Baken,	Technische Universiteit Delft, reservelid

ISBN 90-5335-070-5

Keywords: IP multicast, Application Layer multicast, Shortest Path Tree, Steiner Minimum Tree, traceroute

Copyright © 2005 by M. Janić

All rights reserved. No part of the material protected by this copyright notice may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage and retrieval system, without written permission from the author.

Printed in The Netherlands.

to my parents Milka and Radovan,

Contents

1 Introduction 1
- 1.1 Internet: past, present, future . . . 1
- 1.2 Multicasting: Network (IP) Layer . . . 4
- 1.3 Multicasting: Application Layer . . . 6
- 1.4 Thesis Objectives and Research Questions . . . 7
- 1.5 Thesis Outline . . . 9

2 Trees: Definitions and Properties 11
- 2.1 Shortest Path Tree . . . 13
- 2.2 Steiner Minimum Tree . . . 16
- 2.3 Uniform Recursive Tree (URT) . . . 19

I Multicast on Network (IP) Layer 23

3 IP Multicast: State of the Art 25
- 3.1 Best Current Practice . . . 25
  - 3.1.1 Multicast protocols . . . 25
  - 3.1.2 Multicast service models: Any Source Multicast (ASM), Source Specific Multicast (SSM) and Explicit Multicast . . . 29
- 3.2 Deployment Status . . . 31
  - 3.2.1 Deployment Challenges . . . 34

4 Properties of Multicast Routing Trees 37
- 4.1 The gain of multicast over unicast . . . 37
  - 4.1.1 Gain: Chuang-Sirbu law and log-log scale insensitivity . . . 37
  - 4.1.2 Gain: Theory . . . 39
- 4.2 Stability of Multicast Routing Trees . . . 39
  - 4.2.1 Stability: Theory . . . 40
  - 4.2.2 Stability: Simulation Results . . . 43
  - 4.2.3 Comparison of Steiner and Shortest Path Tree . . . 56

4.3 Cost of Multicast Routing Trees 56
4.3.1 Cost: Simulation Results and a Conjecture 60
4.4 Implications for the Network Providers 65
4.5 Conclusions 67

5 Measurement-Based Analysis of Multicast Trees 71
5.1 Measurement data sets 72
5.1.1 RIPE-NCC 73
5.1.2 CAIDA 75
5.1.3 PlanetLab 76
5.2 Ambiguities in *traceroutes* 77
5.2.1 Errors and inaccuracies 77
5.2.2 Alias resolution 78
5.2.3 Bias sampling 79
5.2.4 Analysis of cycles 80
5.3 Node degree distributions in the maps of Internet 82
5.3.1 Overview of node degree distributions obtained in other research groups 82
5.3.2 Node degree distributions based on CAIDA, RIPE NCC and PlanetLab data 85
5.3.3 Simulations of the union of Shortest Path Trees in the complete graph 88
5.4 The Structure of Multicast Trees 89
5.4.1 Related work 89
5.4.2 Node-degree distribution of trees in RIPE and CAIDA 91
5.4.3 Number of links in CAIDA trees-the gain of multicast 94
5.5 Conclusions 94

II Multicast on the Application Layer 97

6 Overview of AL Multicast Protocols 99
6.1 Mesh-based (mesh-first) multicast 100
6.1.1 Narada 100
6.1.2 ScatterCast 102
6.2 Tree-based (tree-first) multicast 102
6.2.1 Banana Tree Protocol (BTP) 102
6.2.2 Host Multicast Tree Protocol (HMTP) 103
6.2.3 Yoid 105
6.2.4 ALMI 106
6.2.5 Overcast 107

6.2.6 Overlay Tree Building Control Protocol (TBCP) 107
6.2.7 Short general comparison of mesh-based and tree-based protocols 109
6.3 Multicast on top of structured P2P overlays 109
6.3.1 CAN-based Multicast (MCAN) 111
6.3.2 SCRIBE 113
6.3.3 Pastry 113
6.3.4 Bayeux 115
6.3.5 Delaunay triangulation 115
6.3.6 HyperCast 119
6.4 Hierarchical multicast 120
6.4.1 Kudos 121
6.4.2 NICE 122
6.5 Conclusion 124
6.5.1 Efficiency 124
6.5.2 Scalability 125

7 Hopcount in Application Layer Multicast **127**
7.1 Problem statement 127
7.2 Forwarding in CAN based Multicast (MCAN) 128
7.3 Evaluation via Simulations 129
7.3.1 MCAN on a random graph 131
7.3.2 Scribe on a random graph 136
7.3.3 NICE on a random graph 139
7.3.4 Comparative analysis on a random graph 140
7.4 Evaluation via Measurements on PlanetLab 143
7.5 Conclusions
145

8 Conclusions **147**
8.1 Network Layer Multicast 147
8.2 Application Layer Multicast 150
8.3 Thesis Contributions 151

9 Abbreviations **153**

10 Symbols **157**

Bibliography **159**

Summary **169**

Samenvatting (Summary in Dutch) **171**

Acknowledgements **175**

Curriculum Vitae **177**

Chapter 1

Introduction

1.1 Internet: past, present, future

A few technologies so far, if any, have had such a tremendous impact on the society as the Internet did. The Internet, the network of networks, that complex structure of interconnected routers, has its roots in the seventh decade of the previous century. The vision of a network that would connect machines and people worldwide has been first articulated by JCR Licklider, the head of the computer research department at the Advanced Research Projects Agency (ARPA), in 1963. In the mid-1960s, ARPA recognized the need to connect university and government research centers into a nationwide network that would allow a wide variety of computers to exchange information and share resources. By the end of 1969, four host computers were connected together into the initial ARPAnet (Figure 1.1). This network was about to form the foundation of the Internet.

Universities and research organizations were among the first to join that network in order to exchange information. However, many changes needed to be undergone, for Internet to become what it is today. In 1972 electronic mail was introduced, which turned out to be a killer application. The original communications protocol NCP was substituted by a new communications protocol pair – Transmission Control Protocol/Internet Protocol (TCP/IP), created in 1973. It was accepted by the U.S. government in 1978, and became the networking standard in 1983.

More networks began to emerge in the 1980s. Evermore educational and commercial organizations wanted to use the same packet-switching technologies. The system became known as the Internet in this period. It had far exceeded its original purpose, and was providing the impetus for a vast technological revolution.

Additional changes were necessary before the Internet could function as a global information utility. In 1989 the World Wide Web project was proposed, as well as a new language for linked computers known as HTML (Hyper-Text Markup Language).

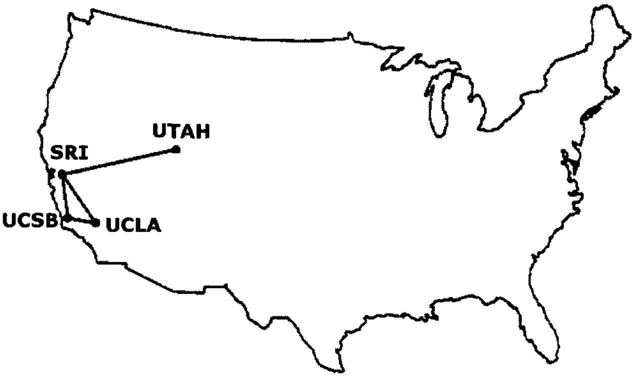


Figure 1.1: ARPAnet, 1969.

Region	Population	Internet Usage	Usage growth (2000 – 2005)	Penetr. (in %)	World Users
Africa	900.465.411	12.937.100	186.6 %	1.4 %	1.6 %
Asia	3.612.363.165	266.742.420	133.4 %	7.4 %	32.6 %
Europe	730.991.138	230.923.361	124.0 %	31.6 %	28.3 %
Middle East	259.499.772	17.325.900	227.8 %	6.7 %	2.1 %
North America	328.387.059	218.400.380	102.0 %	66.5 %	26.7 %
South America / Latin America	546.917.192	55.279.770	205.9 %	10.1 %	6.8 %
Oceania / Australia	33.443.448	15.838.216	107.9 %	47.4 %	1.9 %
World Total	6.412.067.185	817.447.147	126.4 %	12.7 %	100.0 %

Table 1.1: Internet Usage Statistics, February 3, 2005; (source http://www.internetworldstats.com/stats.htm)

Simple tools to retrieve information from the Web and to communicate have become the focus of much activity in the next few years. In the spring of 1993, a group of graduate students at the University of Illinois created a "browser" program called Mosaic, and distributed it free of charge. Netscape, followed by Microsoft, developed browsers that greatly simplified the ability to search the Internet for information.

Today, the Internet has grown into a low-cost technology, a gateway to a vast wealth of knowledge and information, dramatically changing the world. It is available to people at homes and offices, at schools and universities, and in public libraries and "cyber cafes." Some statistics on the penetration of the Internet are given in Table 1.1.

How do the computers on the Internet communicate with each other? They communicate by exchanging the packets containing the desired information. The Internet

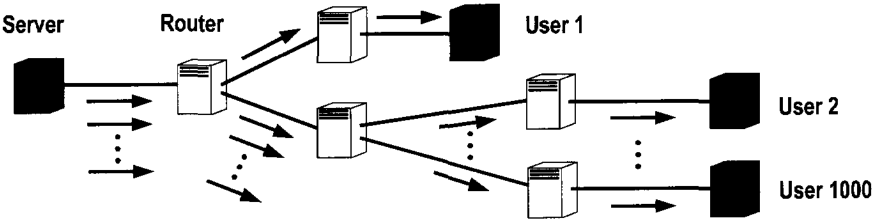


Figure 1.2: Unicasting a movie to 1000 users.

as we know it today is based on unicast (point-to-point) type of communication. In unicast, a packet containing the information is transmitted through the network from a source toward the destination on a hop-by-hop basis. That is, each router on the path between the source and destination selects the next router (hop) for a given packet such that the path between a source and destination contains a minimum number of hops (routers traversed). This shortest path across a network can be found with the use of a Shortest Path algorithm (e.g. Dijkstra's algorithm [33]). Shortest Path algorithms will be explained in more detail in Chapter 2. Most of the traditional IP applications, such as Web browsing and e-mail, employ unicasting. Whenever each user has unique needs - browsing a different Web site, receiving a particular message - unicasting is justified (there is no wasteful duplication of data).

In the last couple of years, we have witnessed the proliferation of multimedia applications that involve simultaneous participation of several users. These applications include one-to-many, and many-to-many type of communications. One-to-many applications have a single sender, and multiple simultaneous receivers. They include scheduled audio/video distribution, push media, file distribution and caching. In many-to-many applications, two or more of the receivers also act as senders. Some applications of this type are multimedia conferencing, concurrent processing, distributed interactive simulations (DIS), multiplayer interactive games, distance learning, and many more.

Most of these applications are unable to run on today's Internet, due to high bandwidth demands. To illustrate this point, let us assume that a video server wants to transmit a movie via unicast to 1000 recipients (Figure 1.2). The server needs to employ 1000 separate point-to-point connections, and to send 1000 copies of the movie over a network. This is apparently a waste of network capacity, as it is the same information that is contained in each of the packets. Moreover, it is a waste of server capacity as well. For example, suppose that a server is offering a live video stream that requires 1 Mb/s for each client. With a 100 Mb/s network card on the server, its interface would be completely saturated after more than 90 clients connected to it. In order to resolve these problems, multicast was invented.

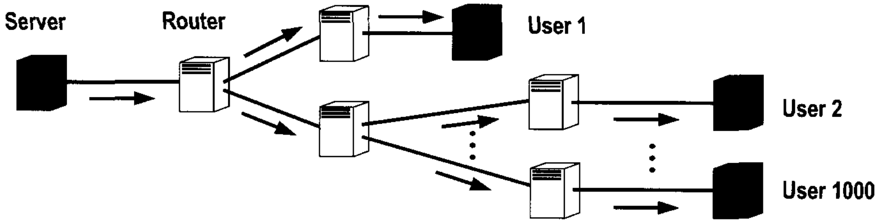


Figure 1.3: Multicasting a movie to 1000 users.

1.2 Multicasting: Network (IP) Layer

Multimedia application deployment can be greatly facilitated by the use of multicast communication. As long as there is an overlap in paths toward several destinations, only one copy of a data packet will be sent over any network link on that path. When paths divert, packets are duplicated in the router and sent out to the appropriate links. As Figure 1.3 illustrates, each packet stream is replicated in the network there where needed.

The benefits of multicast usage are twofold. Multicast reduces the load on the server, since the server has to send only one packet per link, instead of multiple packets to different receivers. In addition, it reduces the overall network load, as only one copy of data packet is transmitted wherever possible. This dramatically reduces the overall bandwidth consumption, as can be intuitively concluded when comparing Figure 1.2 and Figure 1.3.

The standard multicast model has been introduced and described by Steven Deering in 1988 [35]. The Network Layer (IP) multicast model proposed by Deering is based on a notion of a group: hosts that are interested in a particular application form a multicast group. Each multicast group is identified with a special class-D multicast address. To receive data from a multicast group, hosts must join the group by contacting the routers they are attached to, using the Internet Group Management Protocol version 3 (IGMPv3) [21]. Once a host joins a group, it receives all data sent to the group address regardless of the sender's source address. Hosts can send to a multicast group without becoming a receiver; such hosts are often referred to as non-member senders.

The IP multicast group model is an open service model. No mechanism restricts the hosts or users from creating a multicast group, receiving data from a group, or sending data to a group. Multiple senders may share the same multicast address. Senders cannot reserve addresses nor prevent another sender from choosing the same address. The notion of group membership is only a reachability notion for receivers and is not meant to provide any kind of access control. As a result, an IP multicast group is not easily managed.

The multicast routing problem, the problem of finding an optimal (shortest) path, is more complicated than that of unicast. Whereas in unicast a shortest path should be found connecting a source to the destination (a path P_i connects server to user i in Figure 1.2), in order to perform multicast, all nodes belonging to the multicast group must be interconnected by a tree (the server is connected to all users 1 to 1000 in Figure 1.3). Such a tree is called a multicast tree and is indicated in Figure 1.3 with thick arrows. Multicast packets are then forwarded along this tree from the sender to all multicast group members.

Several approaches have been adopted for determining the multicast tree. The simplest way to build a routing tree is to add one participant at a time, using a shortest path algorithm. New participants are connected along a shortest path to the nearest node in the existing tree. While the Shortest Path Tree between the source node and each destination node guarantees that multicast packets will be delivered as fast as possible, it does not necessarily result in a tree that economizes on network resources. The second approach is to construct a single tree to distribute the traffic from all senders in the group, regardless of the sender's location, and to minimize the total weight of the tree. Hence, it optimizes the use of network resources. The problem of finding a minimum weight tree that spans all multicast users is known as the Steiner Tree problem in networks [45]. Both Shortest Path Trees and Steiner Trees will be explained and discussed in more detail in Chapter 2.

Multicast routing trees are created and maintained by a multicast routing protocol. Many such protocols have been proposed to this end. The deployed architecture has however converged toward just a few of them. None of them, for the reasons that will be explained later, exploits Steiner Trees. Instead, they are all based on a version of the Shortest Path Tree [84, 105, 38].

Even though the first multicast deployment took place in 1992, multicast is still not widely deployed. Recent data show that only $3-5\%$ of the total number of Autonomous Systems[1] are capable of providing multicast. Various technical and non-technical issues have stalled its wide-spread use (security, scalable address allocation, reliability, infrastructure, management). For years, those issues have been a subject of research, and some solutions have been proposed. We will discuss most of these issues in Chapter 3.

---

[1] Autonomous System is a collection of networks under a common administration sharing a common routing strategy (Cisco Systems technical definition).

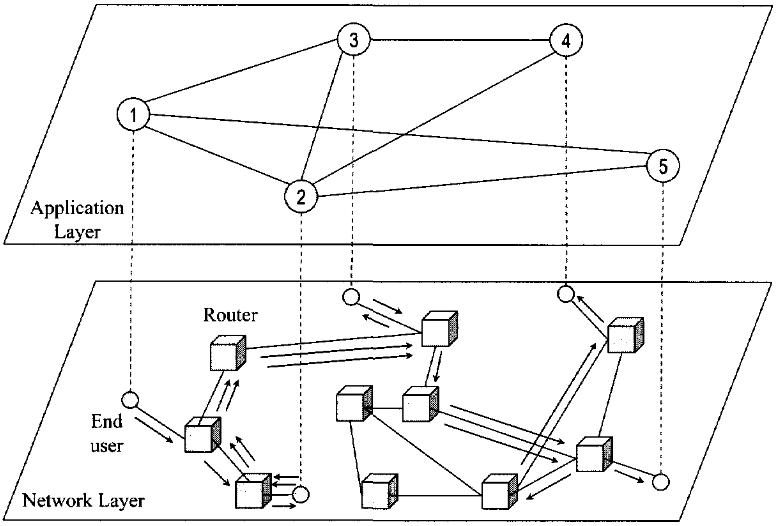


Figure 1.4: The overlay network formed by end users 1 to 5 and the corresponding underlying network-layer infrastructure. The application layer multicast tree is indicated with thick lines. The packets follow unicast paths in the underlying network, as indicated with arrows.

1.3 Multicasting: Application Layer

The most efficient way to provide multicast is to implement it in the network layer[2], as discussed above. However, the slow deployment of IP multicast and the rising need to support multiuser applications have provided impetus for the emergence of Application Layer multicast [109, 112, 86, 31, 11, 40, 12, 68, 110]. In Application Layer (AL) multicast, as the name implies, packet replication and routing are handled by the participating end-hosts, instead of by the network routers as is the case in IP multicast. The end-hosts form an overlay network, in which each overlay link corresponds to a direct unicast path between two group members. All the data packets are sent as unicast packets and forwarded from one member in the overlay network to another according to some defined rules. The concept of Application Layer multicasting is illustrated in Figure 1.4.

As can be seen from Figure 1.4, AL multicast does not require additional network

---

[2] The network layer refers to the network layer (layer 3) in the Open Systems Interconnection (OSI) model. The OSI model defines a networking framework for implementing protocols in seven layers. Control is passed from one layer to the next, starting at the application layer (layer 7) in one station, proceeding to the bottom layer, over the channel to the next station and back up the hierarchy.

support or changes in the infrastructure. This enables rapid and seamless deployment of multicast applications. This triggered some to believe that if AL multicast would become more widely deployed, it might enhance a further creation of multicast applications, which might in turn speed up the deployment of IP Multicast.

1.4 Thesis Objectives and Research Questions

In this thesis, we focus on multicast in both network and application layer. Our research on network layer multicast has been driven by one significant and yet unresolved reason for the slow wide-scale deployment of IP multicast, which is the lack of a business incentive for network providers. IP multicast is namely more complex than unicast. The computational and administrational overhead of multicast group management and routing increases the deployment cost compared to the cost of unicast. Clearly, the deployment of multicast can only be justified if the savings offered by multicast exceed the cost of its deployment. We take the standpoint that the deployment of multicast on larger scale would be encouraged if we could reliably estimate the savings and costs that multicast deployment brings to network providers. In order to establish multicast business scenarios, the savings and costs of multicast over unicast have to be determined. First, they need to be defined. The savings in network resources that multicast provides can be expressed in terms of the number of hops (links among each pair of routers) in the tree rooted at a particular source to m randomly chosen receivers, compared to the total number of hops (links) when the message is separately routed to each of the receivers. The cost of multicast on the other side is composed out of several factors, such as the cost of deployment and management. Once defined, the savings and cost should be determined analytically. In order to do so, the behavior and properties of multicast routing trees must be understood. Finally, the theoretical findings should be supported by measurements on the Internet itself. Hence,

our objective regarding network-layer multicast is

- *To identify, analyze and quantify the factors that impact the cost of multicast over unicast, and to verify them via measurements on the Internet.*

AL multicast has emerged as an interim solution to the deployment problems of IP multicast. The reduced complexity of AL multicast with respect to IP multicast comes however at the expense of efficiency, since in AL multicast packets may traverse the same link several times (as depicted in Figure 1.4). AL multicast can obviously only make sense if it outperforms unicast. Therefore, one of the crucial questions to be answered is "how efficient is AL multicast?" All AL multicast algorithms proposed so far claim to be more efficient than unicast, and slightly less efficient than IP multicast.

These claims are usually supported by few simulation results. In most cases, different algorithms are not compared under the equal conditions, which complicates drawing conclusions. We believe that understanding the performance of existing AL multicast protocols will teach us how to optimize and improve the best among them.

One critical issue for efficient application layer routing is the quality of the match between the overlay and the underlying network topology. Namely, the end nodes do not possess the topological information available to routers. Hence, when connecting to each other into an overlay, often they do not take the underlying topology into account. As a result, the two neighboring nodes in the overlay may be separated by many hops on the IP layer, leading to high delays and unnecessary traffic. Overlay performance, in terms of delay penalty and the number of duplicate packets, could be improved if the connectivity between the nodes in the overlay would be congruent with their connectivity in the underlying network.

Gathering topological information in a manner that is both practical and scalable is not a trivial task. In addition to gathering, the effective incorporation of this information into the design of overlay networks is just as challenging. Several methods for obtaining and incorporating topological information have been developed thus far. Nevertheless, none of them leads to a complete congruence of overlay with its underlying substrate. Some researchers argue that AL multicast applications do not require exact topological information and can instead use sufficiently informative hints about the relative positions of Internet hosts. Triggered by this statement, we investigate the influence that the knowledge of topology has to AL multicast performance. We consider two extreme cases. In one extreme the end users possess no information on the underlying topology, and they are connected to each other in a random manner. The other extreme is the optimal situation in which the complete knowledge of the underlying substrate is attained and used by the end users for the creation of the overlay (or for routing). In both scenarios we compare the efficiency of AL multicast schemes to IP multicast, unicast and mutually among the schemes themselves. To summarize,

our objectives with respect to Application Layer multicast are

- *To evaluate the efficiency of three prominent application layer multicast algorithms.*
- *To investigate the impact the topology awareness has on the efficiency of application layer multicast algorithms.*

1.5 Thesis Outline

The schematic overview of this thesis is given in Figure 1.5. This thesis is organized in two parts. The first part is dedicated to Network (IP) Layer multicast, whereas in the second part, the focus is on Application Layer multicast. Prior to Part I, there are two introductory chapters. Chapter 1 introduces and discusses different ways of performing multicast communication, as well as our motivation and research objectives.

Chapter 2 contains some background information needed to follow the material discussed in the rest of the thesis. It covers the basic definitions of graph theory and further focuses on different algorithms used for building the multicast routing trees, namely Shortest Path Tree and Steiner Minimum Tree. It also introduces and defines the Uniform Recursive Tree.

Part I starts with Chapter 3, which explains the mechanisms of IP multicast in more detail. Further, Chapter 3 discusses the current IP multicast "best practice" architecture, as well as the deployment problems and challenges.

Chapter 4 analyzes the properties of multicast routing trees. Understanding and quantifying these properties, such as the gain, the cost and the stability of multicast routing trees, has a direct impact on the multicast business scenario. Implications of our theoretical findings for the business model are covered in the same chapter.

Network providers can only be persuaded to consider the business model that is based on the mathematical expressions we propose if they hold for the graph of Internet as well. Therefore, in Chapter 5 we verify our theoretical results on multicast trees via extensive Internet measurements. In addition to multicast trees, maps of Internet have been analyzed, based on various measurements and measurement sources. The same chapter discusses the significant issues regarding Internet *traceroute*-based sampling, and ambiguities that emerge from these methods.

Chapter 6 is the first chapter of Part II, and it presents a classification and a detailed overview of seventeen different Application Layer multicast protocols. From them, we analyze the three that we believe have most potential, in terms of scalability and efficiency. The results of our analyses of efficiency and sensitivity to underlying topology are presented and discussed in Chapter 7.

Finally, Chapter 8 articulates the main conclusions and outlines the thesis contributions.

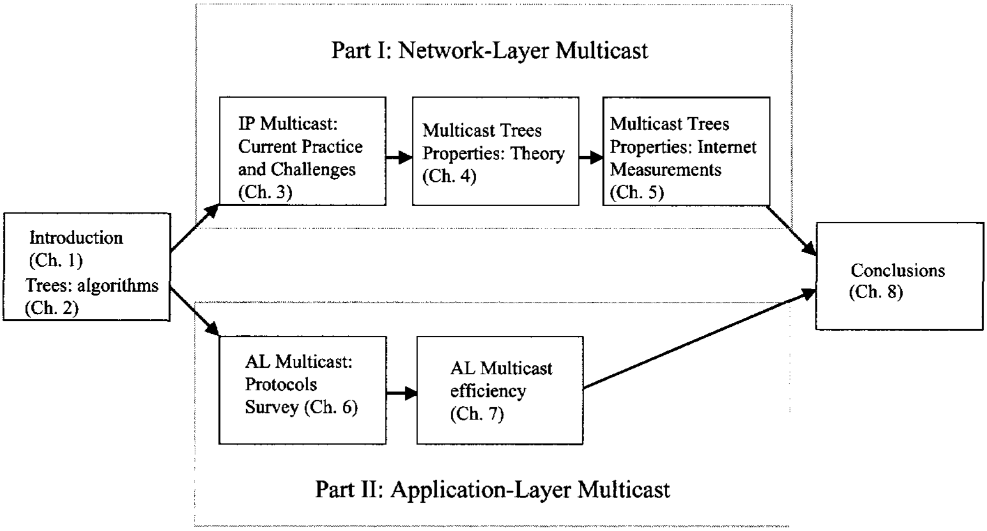

Part I: Network-Layer Multicast
IP Multicast: Current Practice and Challenges (Ch. 3)
Multicast Trees Properties: Theory (Ch. 4)
Multicast Trees Properties: Internet Measurements (Ch. 5)
Introduction (Ch. 1) Trees: algorithms (Ch. 2)
Conclusions (Ch. 8)
AL Multicast: Protocols Survey (Ch. 6)
AL Multicast efficiency (Ch. 7)
Part II: Application-Layer Multicast

Figure 1.5: Thesis outline.

Chapter 2

Trees: Definitions and Properties

Each (communication) network can be viewed as a set of nodes connected with a set of links. Therefore, each network topology can be represented in the form of a graph. A graph $G(V, E)$ is a structure consisting of a set of vertices V and set of edges E connecting them. In networking terminology, a set of vertices is referred to as a set of nodes $\mathcal{N} = \{1...N\}$, and a set of edges as a set of links $\mathcal{L} = \{1...L\}$, where $N = \dim(\mathcal{N})$ and $L = \dim(\mathcal{L})$ are the number of nodes and links in the network, respectively. Hence, instead of $G(V, E)$, the notation $G(N, L)$ is used. Graphs can be visually depicted as a set of points (nodes) connected by lines (links) (see Figure 2.1). In this chapter we provide some basic theory on graphs and algorithms, needed for a better comprehension of the rest of material treated in this thesis.

We begin by providing several definitions that we will rely on and refer to in the sequel. These definitions follow the notation given in the book of Harary [54].

Definition 1 *Walk: A walk from node u to node v is an alternating finite sequence $n_0, l_1, n_1, \ldots l_k, n_k$, of nodes n_i and links l_i, where l_i is a link connecting node n_{i-1} and n_i,*

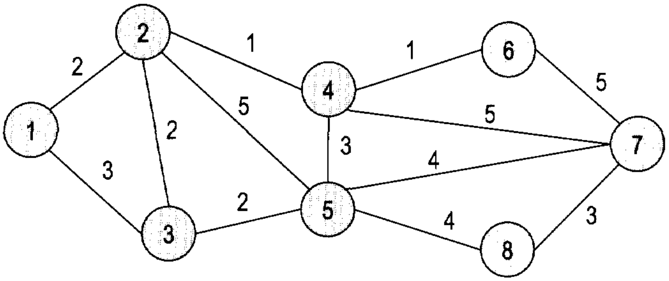


Figure 2.1: An example of a graph with $N = 8$ nodes and $L = 13$ links.

$n_0 = u$ and $n_k = v$.

Definition 2 *Path: A path is a walk in which all nodes n_0 to n_k are distinct ($n_i \neq n_j$ for every index i, j)*

Definition 3 *Connected: A graph is connected if there exists a path between each pair of nodes in the graph.*

Definition 4 *Degree: A degree $\deg_n$ of a node n represents the number of nodes adjacent to node n (the number of nodes to which node n is directly connected with a link).*

Definition 5 *Cycle: A cycle is a walk for which all nodes except the first and last are distinct. If a graph contains no cycles it is called acyclic.*

Definition 6 *Tree: A tree is a connected acyclic graph.*

Graphs can be represented in the form of an adjacency matrix, or alternatively, adjacency list. The adjacency matrix A is a $N \times N$ matrix, whose elements a_{ij} are filled in the following way: if there is a link between nodes i and j, element a_{ij} will take a value of 1, otherwise $a_{ij} = 0$. An adjacency matrix is an efficient way of graph representation from the memory point of view in case of dense graphs (large number of links). For sparse graphs, adjacency lists should be used instead.

A graph is however not only defined by its topological structure, the link weight information is just as relevant. One common mistake many network researchers make when modeling communication networks is neglecting the fact that not all the links in the network are the same. Some links have higher capacity than others, some links are longer than others and some links are more expensive than others. Hence, in addition to the node connectivity information embodied in the adjacency matrix/adjacency list, the information on the diversity of links must be included as well. This can be achieved by associating a weight vector $\vec{w}(i \rightarrow j)$ to each network link. A link weight vector $\vec{w}(i \rightarrow j)$ contains l positive elements, each of which reflecting a QoS measure, such as delay, available capacity, physical distance, cost, jitter, etc. In the sequel, we confine to the single link weight case, with only one additive link metric ($l = 1$). If each element $a_{ij} \neq 0$ in the adjacency matrix A is substituted by its corresponding link weight value $w(i \rightarrow j)$, a topology matrix T is obtained.

Often symmetry in both directions is assumed, i.e. $w(i \rightarrow j) = w(j \rightarrow i)$, leading to undirected graphs, where each pair of nodes linked in one direction is also linked in the other. By convention, links in the undirected graph are represented with a line without an arrow, as illustrated in Figure 2.1. This assumption seems trivial, however, in telecommunications, the up-link and down-link transfer of information is usually asymmetrical. In [79], Paxson has discovered that half of the paths in the Internet (in 1995) from a source A to a destination B have been different than paths from B to A.

Some classes of graphs that will be treated in this thesis are:

1. A random graph: A random graph, denoted by $G_p(N)$, is a graph consisting of N nodes, where the probability that any two nodes are connected equals p. The class of random graphs has been studied in detail by Bollobas [18] and later by Janson *et al.* [63]. We use the term RGU to refer to the random graphs $G_p(N)$ with uniformly on $[0, 1]$ (or exponentially) distributed link weights w.
   One of the characteristics of the random graph is that its node degree distribution (i.e. the probability that each node has i adjacent nodes), is a binomial, $\Pr[\deg = i] = \binom{N-1}{i} p^i (1-p)^{N-1-i}$, with the average degree $d_a = p(N-1)$. For large N, the binomial distribution can be represented as a Poisson distribution in the form $\frac{e^{-d_a} d_a^i}{i!}$.

2. A complete graph: A complete graph is attained when $p = 1$ in the random graph. In that case each node is connected to all the other nodes in the graph. A complete graph consisting of N nodes (and consequently of $\binom{N}{2}$ links) is denoted as K_N.

Our interest in random graphs lies in the fact that they represent a good model for some types of realistic networks, such as Ad-Hoc wireless networks [55] and certain peer-to-peer networks. However, as we will discuss in Chapter 5, the topology of Internet seems not to be one of them. From measurements in the Internet, it has been observed [46] that the node degrees of the Internet nodes are not binomially, but polynomially distributed, i.e. $\Pr[\deg = i] = ci^{-\alpha_\tau}$, where c is a constant such that $\sum_{i=1}^{N-1} ci^{-\alpha_\tau} = 1$. Nevertheless, in Chapter 5 we will demonstrate that these conclusions for the topology of Internet have been drawn based on unreliable data. Moreover, in the same chapter we will further show that the class of random graphs, although not a good model for the Internet itself, seems to represent a reasonable model for multicast trees in the Internet.

If we assume that a graph models a telecommunication network, a next question that arises is how a message in such a network is routed, i.e. which path the message follows from a source to the destination. This problem is known as the routing problem in telecommunication networks. In case of unicast, the routing problem is translated to a problem of finding a shortest path between a source and a destination. In multicast however, there are multiple destinations, hence a message should be delivered along the tree, spanning all the multicast members interested in the particular message.

In the remainder of this chapter we will describe different algorithms for building multicast routing trees, and we will briefly discuss some properties of these trees.

2.1 Shortest Path Tree

The Shortest Path Tree spanning a source and m destinations, is a tree rooted at the source, and composed as the union of shortest paths between the source and each of the destinations. Hence, the tree constructing problem reduces to finding the shortest

```
DIJKSTRA$(G, s, t)$
1. INITIALIZE$(G, s, \delta, \pi)$
2. queue $Q \longleftarrow \mathcal{N}$
3. while $Q \neq 0$
4.     EXTRACT-MIN$(Q) \rightarrow u$
5.     if $u = t$
6.         stop and return path
7.     else
8.         for each $v \in adj[u]$
9.             RELAX$(u, v, w, \delta, \pi)$
```

Figure 2.2: *The pseudocode of the Dijkstra shortest path algorithm.*

paths between the source and each of the destinations respectively, and then assembling them together. The Shortest Path Tree for the graph depicted in Figure 2.1 is shown in Figure 2.6 (node 1 is the source, nodes 5, 6 and 7 are destinations).

The shortest path problem can be rewritten formally in the following way: Consider a directed and strongly connected graph G with N nodes and L links, where each link between node i and a node j is specified by a link weight $w(i \rightarrow j)$. Assume further that the single link weight is additive, which means that the weight of a path $P = n_0 \rightarrow n_1, \rightarrow \ldots \rightarrow n_k$, equals the sum of the weights of its constituent links:

$$w(P) = \sum_{j=1}^{k-1} w(n_j \rightarrow n_{j+1})$$

The shortest path between node s and t is such a path $P^*_{s \rightarrow t}$ that minimizes $w(P^*_{s \rightarrow t})$, i.e. for any path $P_{s \rightarrow t}$ it holds that $w(P^*_{s \rightarrow t}) \leq w(P_{s \rightarrow t})$.

Several algorithms [14][64] for the discovery of the shortest paths have been proposed. Pioneering work has been performed by Edsger Wybe Dijkstra, who in 1959 proposed an algorithm today referred to as the Dijkstra algorithm [37]. It is a greedy algorithm that relies on both the principle of relaxation and the fact that the subsections of shortest paths are shortest paths. This algorithm associates an "attribute" $\delta(v)$ to each node v in the graph G. At each step, the value of the attribute is the value of the sum of the weights on the shortest path from source s to node v that far. Further, the Dijkstra algorithm maintains the list of predecessor nodes $\pi(v)$, which is either another node, its predecessor in the path backward to the source, or NIL. The pseudocode of the Dijkstra algorithm is given in Figure 2.2.

In line 1 the initialization of attributes $\delta[v]$ and predecessors $\pi[v]$ for all the nodes v in the graph is performed, according to INITIALIZE procedure shown in Figure 2.3. As can be seen from Figure 2.3, for all nodes in the graph the values of attributes are set

```
INITIALIZE($G, s, \delta, \pi$)
1. for $v \in \mathcal{N}$
2.     $\delta[v] \leftarrow \infty$
3.     $\pi[v] \leftarrow NIL$
4. $\delta[s] \leftarrow 0$
```

Figure 2.3: The Initialization procedure.

```
RELAX($u, v, w, \delta, \pi$)
1. if $\delta[v] > \delta[v] + w(u \rightarrow v)$
2.     $\delta[v] \leftarrow \delta[v] + w(u \rightarrow v)$
3.     $\pi[v] \leftarrow u$
```

Figure 2.4: The Relaxation procedure.

to infinity (line 2), and the values of predecessors to NIL (line 3). Only the attribute $\delta[s]$ of the source s is set to zero in line 4. Line 2 in the Dijkstra algorithm inserts all nodes of the graph G to the queue Q. The main algorithm begins with line 3. In line 4 node u with the shortest weight (i.e. $\delta(u) < \delta(v), v \in Q, v \neq u$) is found in the queue and removed from it. Node u can be seen as a new searching node toward the destination t. Line 5 verifies if the searching node u is the same as the destination t, in case of which the algorithm stops and returns the path (line 6). Lines 7 to 9 in Dijkstra perform for each neighbor v of the searching node u the relaxation process, described in Figure 2.4. Line 1 in RELAX procedure verifies whether the current distance $\delta[v]$ from source s to node v can be decreased by reaching node v through the searching node u. In that case, the value of attribute $\delta[v]$ is updated to the new, decreased value (line 2 in RELAX). Also, the predecessor of node v is updated to u (line 3 in RELAX). The shortest path returned in line 6 is represented with a predecessor list π running backwards from the destination node t to the source node s. The algorithm stops when either queue Q is empty (the shortest paths to all the other nodes have been found), or when the searching node u is equal to the destination node t.

It should be noted that lines 5 and 6 are optional, they are used only when a path between source s and a single destination t needs to be found. The original Dijkstra algorithm does not include lines 5 and 6, and it returns the shortest paths to all the other nodes in the graph G, i.e. the shortest path tree rooted at source s. Similarly, the Dijkstra algorithm can be used for finding a shortest path tree from source s to a set of m destinations $\mathcal{T}$. In that case, lines 5 and 6 in DIJKSTRA need to be substituted with lines $5a$ to $9a$, given in Figure 2.5. Line $5a$ checks whether the searching node u is

```
5a.  if $u \in \mathcal{T}$
6a.      if $dest\_found = m - 1$
7a.          stop and return paths
8a.      else
9a.          $dest\_found = dest\_found + 1$
```

Figure 2.5: The modifications of the Dijkstra algorithm.

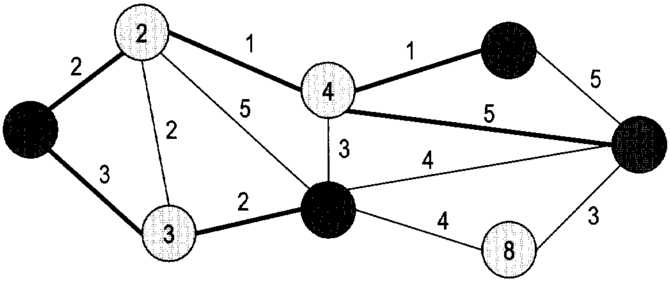


Figure 2.6: Shortest Path Tree connecting node 1 to nodes 5, 6 and 7.

the same as any of the m destinations from the destination set $\mathcal{T}$. If so, the algorithm checks in line 6a whether all the other $m-1$ destinations have been found previously, in which case the algorithm stops and returns the paths in line 7a. Otherwise, if the searching node u is the same as one of the destination nodes, but it is not the last destination node to be found, the counter *dest_found*, that maintains the number of destinations found that far, needs to be incremented. The algorithm further proceeds in the same manner as the original one (line 9a is followed by lines 7 to 9 in Figure 2.2). In addition to these modifications, the counter *dest_found* has to be initialized to the value of zero in INITIALIZATION procedure.

The worst-case complexity of the Dijkstra algorithm (when using Fibonacci heaps) equals $C_{Dijkstra} = O\left(N \log N + L\right)$. The proof that the Dijkstra algorithm is exact can be found in [33].

2.2 Steiner Minimum Tree

The classical optimization problem in multicast routing is called the Steiner Tree problem in networks, and can be formulated as follows: given a graph $G = (N, L)$, with link weights $w(i \to j)$, and a non-empty set $\mathcal{R}$ ($\mathcal{R} \subset \mathcal{N}$) of required nodes (in the multicast routing problem this set is made of $m_t = \dim(\mathcal{R})$ multicast group members), find an acyclic subgraph $\mathcal{T}_G(m_t)$ of G that contains a path between every pair of nodes and minimizes $w(\mathcal{T}_G(m_t))$, where $w(\mathcal{T}_G(m_t))$ is defined as

$$w(\mathcal{T}_G(m_t)) = \sum_{(i \to j) \in \mathcal{T}_G(m_t)} w(i \to j)$$

The difficulty with the Steiner Tree problem is that its determination belongs to the class of NP-complete problems. A problem Π is defined to be NP-complete if

1. $\Pi \in NP$, and
2. $\Pi' \leq_p \Pi$ for every $\Pi' \in NP$,

Informally, a problem Π is defined to be NP-complete if its solution can be checked/verified by a polynomial-time algorithm[1] (condition 1), and if any other NP-complete problem Π' can be reduced to the problem Π in polynomial time (condition 2). A problem that satisfies condition 2, but not necessarily condition 1 is defined to be NP-hard. That the Steiner Tree problem belongs to the class of NP-complete has been proven by Richard Karp in [65][2]. Condition 2 implies that if any NP-complete problem could be solved in polynomial time, then all NP-complete problems could be solved in polynomial time. However, no polynomial-time algorithm has been discovered for any NP-complete problem so far.

For the Steiner tree problem however, in two special cases, where $m_t = \dim(\mathcal{R}) = 2$ and $m_t = \dim(\mathcal{R}) = N$, there does exist a polynomial time algorithm:

1. $m_t = 2$: The Steiner tree problem reduces to the shortest path problem, for which the solution can be found in polynomial time, with e.g. the Dijkstra algorithm (with the complexity $C_{Dijkstra} = O(N \log N + L)$).
2. $m_t = N$: Since all the nodes in the graph form the group, the Steiner tree problem is translated to the minimum spanning tree problem: find an acyclic subgraph $\mathcal{T}_G(N)$ that connects all nodes in graph G and whose total weight $w(\mathcal{T})$

$$w(\mathcal{T}) = \sum_{(i \to j) \in \mathcal{T}} w(i \to j)$$

---

[1] An algorithms is called a polynomial-time algorithm if it terminates after a number of computational steps bounded by a polynomial in the input size.

[2] NP-hardness (condition 2) has been proven by reducing another NP-complete problem, in this case the Exact-Cover problem, to the Steiner Tree problem.

```
PRIM($G, s$)
1. INITIALIZE($G, s, \delta, \pi$)
2. queue $Q \longleftarrow \mathcal{N}$
3. while $Q \neq 0$
4.     EXTRACT-MIN($Q$)$\rightarrow u$
5.         for each $v \in (adj\,[u]$ and $Q)$
6.             if $\delta[v] > w(u \rightarrow v)$
7.                 then $\delta[v] \rightarrow w(u \rightarrow v)$
8.                      $\pi[v] \rightarrow u$
```

Figure 2.7: The pseudocode of the Prim Minimum Spanning Tree algorithm.

is minimized.

Since $\mathcal{T}$ contains no loops and connects all nodes, it forms a tree, a so called spanning tree, spanning the graph G.

There exist two greedy algorithms to obtain the minimum spanning tree: Kruskal's and Prim's algorithm. Kruskal's [67] algorithm first arranges all links in the increasing order of the link weight. Then, starting with a minimum weight link, it grows a set of partial minimum spanning trees, until all nodes in graph G are connected. Prim's algorithm [82] builds upon a single partial minimum spanning tree, rooted at an arbitrary node s. The pseudocode of the Prim algorithm is given in Figure 2.7. The Prim algorithm operates similarly to the Dijkstra shortest-path algorithm. The first 4 lines are the same as in the Dijkstra algorithm. Line 5 adds an extra condition for every node v that is a neighbor of the searching node u: if node v is not in the queue Q anymore (i.e. it already belongs to the tree rooted at s), it should not be updated. Lines $6-8$ describe the updating procedure, similar to the relaxation process in the Dijkstra algorithm. The difference is in the attribute $\delta[v]$, which contains not the smallest value of the path $w(P_{s \rightarrow v})$ from source s to node v that far, but the smallest link weight that connects v to the tree at node u.

Both Kruskal's and Prim's algorithm have complexity that grows logarithmically with N. Precisely, the complexity of Kruskal's algorithm is $C_{Kruskal} = O\left(L \log N\right)$ and that of Prim's algorithm is $C_{Prim} = O\left(L + N \log N\right)$.

For $2 < m_t < N$ the Steiner Tree problem complicates significantly, since there exists a large number of ways to construct the minimum weight tree, considering this tree can also include none, one, or more (maximum of $N - m_t$) auxiliary nodes belonging to the set $\mathcal{N} \backslash \mathcal{R}$ (the so-called Steiner points; in multicast routing problem those nodes that are not the members of the group). A Steiner minimum tree can be constructed in the following way: consider the set A_k of k auxiliary nodes and the set of group members $\mathcal{R}$. A number of possible ways to choose the set of auxiliary nodes is equal to $N_{G_k} =$

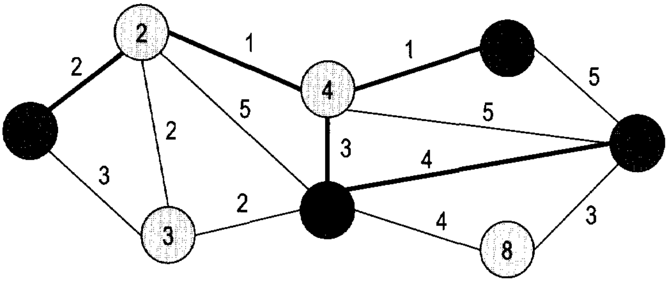


Figure 2.8: Steiner Minimum Tree - the minimum weight tree connecting the multicast group members 1, 5, 6 and 7.

$\sum_{k=0}^{N-m_t}\binom{N-m_t}{k} = 2^{N-m_t}$. The subgraph G_k is obtained from G by removing all other nodes $\mathcal{N}\backslash\{\mathcal{R}\cup A_k\}$ and the links incident to this set of nodes $\mathcal{N}\backslash\{\mathcal{R}\cup A_k\}$. In that subgraph, the minimum spanning tree with the weight $w(\mathcal{T}_{G_k})$ is computed, either with the Kruskal or the Prim algorithm. This procedure is repeated for each of N_{G_k} possible subgraphs G_k. Among all N_{G_k} minimum spanning trees, the one with the minimal value is the Steiner Minimum Tree, $w(\mathcal{T}_G) = \min_{G_k}\left(w\left(T_{G_k}\right)\right)$. Hence, the complexity is approximately $C_{Steiner} = 2^{N-m_t}C_{\Pr im} = O\left(N^2 2^{N-m_t}\right)$. This exponentially fast growing computational complexity of Steiner Trees limits their practical applicability. Even when reductions in the topology, introduced and described in [45], are implemented, the computational time remains unacceptably high.

The Steiner Minimum Tree for a graph in Figure 2.1 is illustrated in Figure 2.6. Shadowed nodes $1, 5, 6$ and 7 represent the multicast users. Nodes 2 and 4 are the so-called Steiner points.

2.3 Uniform Recursive Tree (URT)

The structure to be introduced in this section is called a Uniform Recursive Tree (URT)[92]. A URT is a random tree rooted at a source, that grows according to the following rule: given a URT with N^* nodes, a URT with N^*+1 nodes is deduced by attaching the N^*+1-th node uniformly (thus with probability $1/N^*$) to any of the N^* other nodes in the tree. Our interest in URT stems from the fact that the Shortest Path Tree in a complete graph K_N with exponentially (or equivalently uniformly) distributed weights is exactly, and in a random graph $G_p(N)$ asymptotically, the Uniform Recursive Tree, as is proven in [100]. Therefore, in our analysis of Shortest Path Trees in Chapter 4 and Chapter 5, we will use some of the relevant results for properties of URT. Below, we list analytical expressions for properties of URT of interest for our study.

If the number of nodes is N, the probability distribution function of the hopcount H_N of the path between two arbitrary nodes in the Uniform Recursive Tree is shown [100] to obey precisely

$$\Pr\left[H_N = k\right] = \frac{(-1)^{N-(k+1)} S_N^{(k+1)}}{N!} \tag{2.1}$$

where $S_N^{(k)}$ are the Stirling numbers of the first kind. The average and variance of hopcount H_N obey

$$E\left[H_N\right] = \psi\left(N+1\right) + \gamma - 1 \tag{2.2}$$

and

$$var\left[H_N\right] = \psi\left(N+1\right) + \gamma - \frac{\pi^2}{6} + \psi'\left(N+1\right) \tag{2.3}$$

respectively, where $\psi\left(z\right) = \frac{\Gamma'(z)}{\Gamma(z)}$ is the digamma function [7], and γ is the Euler constant $\gamma = 0.57721...$

Using the asymptotic formulae for the digamma function leads to

$$E\left[H_N\right] = \log N + \gamma - 1 + O\left(\frac{1}{N}\right) \tag{2.4}$$

$$var\left[H_N\right] = \log N + \gamma - \frac{\pi^2}{6} + O\left(\frac{1}{N}\right) \tag{2.5}$$

If $N \to \infty$, the probability distribution (2.1) becomes

$$\Pr\left[h_N = k\right] = \frac{1 + O(\frac{1}{N})}{N} \sum_{m=0}^{k} c_{m+1} \frac{\log N}{(k-m)!} \tag{2.6}$$

where c_k are the Taylor coefficients of $\frac{1}{\Gamma(x)}$ listed in [7]. The above listed formulae will be referred to in the discussion of the gain of multicast over unicast (in terms of the number of used links) in Chapter 4.

The average number of hops from a source to m randomly chosen destinations (thus in a tree connecting the source to m receivers), for every N and m is given by

$$E\left[H_N\left(m\right)\right] = \frac{mN}{N-m} \sum_{k=m+1}^{N} \frac{1}{k} \tag{2.7}$$

Moreover, in [99] the exact probability generating function and probability distribution $\Pr\left[H_N\left(m\right) = k\right]$ have been derived, from which the variance follows as

$$var\left[H_N\left(m\right)\right] = \frac{N-1+m}{N+1-m} E\left[H_N\left(m\right)\right] - \frac{\left(E\left[H_N\left(m\right)\right]\right)^2}{\left(N+1-m\right)} - \frac{m^2 N^2}{\left(N-m\right)\left(N+1-m\right)} \sum_{k=m+1}^{N} \frac{1}{k^2} \tag{2.8}$$

Expressions (2.7) and (2.8) will also be used in Chapter 4, as well as when discussing multicast trees in the Internet in Chapter 5.

Van der Hofstad *et al.* [99] derived further the average $E\left[W_N\left(m\right)\right]$ of the sum of the weights $W_N\left(m\right)$ in the Shortest Path Tree to m uniform destinations in the random graph $G_p\left(N\right)$ with exponentially distributed link weights,

$$E\left[W_N\left(m\right)\right] = \sum_{j=1}^{m} \frac{1}{N-j} \sum_{k=j}^{N-1} \frac{1}{k} \leq \frac{\pi^2}{6} = \zeta(2) \tag{2.9}$$

where $\zeta(z)$ is a Riemann Zeta function. This result will be used in our analysis of cost of multicast in terms of used network resources, in Chapter 4.

Finally, the ratio of the average of the number of nodes with degree k, denoted by D_k^N, over the total number of nodes in the URT has been shown [92] to obey for large N

$$\frac{E\left[D_k^N\right]}{N} = \frac{1}{2^k} + O\left(\frac{\log^{k-1} N}{N^2}\right) \tag{2.10}$$

which is, for large N, close to $\Pr[deg = k]$, the probability that an arbitrary node has a degree k. Hence, the decay rate of the node degree distribution in the URT equals $-\ln 2 = 0.693$. We will use this result in our study of multicast trees in the Internet, in Chapter 5.

Part I

Multicast on Network (IP) Layer

Chapter 3

IP Multicast: State of the Art

The standard IP multicast model, as we already have mentioned in Chapter 1, has been introduced by Steven Deering in 1988 [35] and defined in RFC 1112 [34]. Since the introduction of that basic IP multicast model, other IP multicast models have appeared, with the goal of overcoming the complexities and flaws of the original model. This chapter introduces and briefly describes three different multicast service models: the traditional Any Source Multicast model, Source Specific Multicast, and Explicit Multicast.

Not only the service models, but many different multicast protocols have appeared and evolved in the meantime as well. Current best practice for network-layer (IP) multicast service provision comprises four different protocols: Internet Group Management Protocol (IGMP) [48] [21], Protocol Independent Multicast (Sparse Mode) (PIM-SM) [44], Border Gateway Protocol with multiprotocol extensions [13], and the Multicast Source Discovery Protocol (MSDP) [47]. In this chapter we briefly outline how these protocols work, and how they work together to provide the end-to-end IP multicast service. For more information on the existing multicast protocols we refer to a nice survey by Maria Ramalho [84]. Finally, this chapter lists and discusses the issues that led to a much slower deployment of IP multicast than initially predicted.

3.1 Best Current Practice

3.1.1 Multicast protocols

There are two main actions that have to take place for multicast to be realized. First, the hosts that are interested in a particular multicast group need to inform the routers that they are attached to that they want to join the group. They do this with the use of Internet Group Membership Protocol (IGMP [48] [21]). The multicast-enabled routers in the Internet use the information on location of groups or of senders to determine the delivery tree along which packets are sent from the source to the set of receivers. This delivery tree is determined with the use of multicast routing protocol.

Internet Group Membership Protocol (IGMP)

In order to receive multicast, a host needs to join the multicast groups it wishes to receive from. The Internet Group Management Protocol was designed for hosts to notify the routers that they are attached to that they want to receive traffic on a particular IP multicast group. If a host only wants to send multicast traffic, no protocol between host and router is needed. The host simply sends the multicast packets out on the link, and they will be forwarded by a multicast router. IGMPvl [34] was proposed in conjunction with DVMRP [103], the first multicast routing protocol. IGMPv2 [48] is a current IETF standard. It enables a fast termination of group subscriptions, by the introduction of "leave" messages. IGMPv2 [48] supports the traditional multicast model, Any Source Multicast (ASM). IGMPv3 [21] addresses some of the weaknesses of the ASM model (the ASM model and its weaknesses will be described in Section 3.1.2). IGMPv3 supports both the ASM and the Source Specific Multicast (SSM) model (also to be described in Section 3.1.2), as it allows receivers to subscribe to only specific sources of a particular multicast group.

Multicast Routing Protocols

Based on the information on the location of group members, multicast routing protocols build multicast routing trees. The trees that multicast protocols build can be classified in two groups: source-specific and group-shared trees. A source-specific tree is a tree with the source as a root of the tree and with the branches of the tree that are formed of the paths toward the receivers. This implies that for each individual source sending to each group there exists a separate tree. Because this tree is built as the union of shortest paths through the network, it is also referred to as a Shortest Path Tree (we have discussed Shortest Path Tree in Chapter 2). These trees are used in the one-to-many type of applications, where one source is emitting data to many receivers. In case of group-shared trees, there is only one tree shared by all sources in the group. Steiner Trees, that have also been discussed in Chapter 2, are the optimal group-shared trees. However, due to their complexity and alleged instability, Steiner Trees are not implemented in routing protocols. Instead, group-shared trees use a single common root placed at some chosen point in the network. This shared root is called the rendezvous point (RP). When using a shared tree, sources must send their traffic to the root (RP), and then the traffic is forwarded down the shared tree to reach all receivers. Group-shared trees thus have longer delays (packets must first be sent to the RP before they can be distributed), but require less router state to be maintained.

The first multicast routing protocol to have emerged has been DVMRP [103].

DVMRP builds a source-specific tree and uses a technique known as Reverse Path Forwarding (RPF). RPF is an optimized form of flooding, where the router accepts a packet from source S through interface I only if I is the interface the router would use to reach S. When a node accepts a packet, it forwards that packet to all the interfaces except the one to which the accepted packet arrived. DVMRP implements its own unicast routing protocol in order to determine which interface leads back to the packet source. This technique dramatically decreases the overhead associated with standard flooding. A more refined version of DVMRP includes pruning to flooding: if there are no members in a certain subtree, this subtree is cut off by a "prune" message to the previous router. DVMRP will periodically reflood in order to reach any new hosts that want to receive a particular group. Because of the way the tree is constructed by DVMRP, it is called a reverse Shortest Path Tree.

A protocol by far most used today is Protocol-Independent Multicast (PIM) [36, 44]. PIM supports two different group membership models: dense (a large group size) and sparse (a small group size). The dense mode PIM (PIM-DM) [36] is very similar to DVMRP. It creates source-specific trees, and it uses reverse path forwarding with pruning. The major difference with DVMRP is that PIM relies on the unicast routing tables to retrieve the path back to the source and that it is independent of the specifics of any unicast routing protocol, as its name implies. The sparse mode PIM (PIM-SM) [44] creates group-shared trees, and employs the notion of a rendezvous point (RP) as a root of the multicast tree. Each group has a single RP. When a user wants to join the group, it sends a join message to the RP. This message is processed by all intermediate routers that create an entry in their multicast routing tables as well, forming a branch between the new user and the RP. Each user that wants to multicast a packet, sends the packet toward the RP, encapsulated in a unicast packet. The RP decapsulates the packet and forwards it along the created tree. The major difference between PIM-SM and other shared-tree protocols is that if the data rate of the source exceeds a certain threshold, a source-specific tree can be used in PIM-SM instead of a RP shared tree. The router that the receiver is attached to sends in that case a "join" packet toward the source and a "prune" toward the RP. The source will continue to send a copy of its packets to the RP, considering that there might still exist members in the group that are receiving packets via the RP rooted tree. PIM-SM includes both ASM and SSM functionality: PIM-SSM forms a subset of PIM-SM. PIM-SSM builds shortest path trees rooted at the source immediately and it bypasses the RP connection stage through shared distribution trees. This is because in SSM the router closest to the interested receiver host is informed of the unicast IP address of the source for the multicast traffic.

Inter-domain Multicast Solutions

All the protocols mentioned and described so far are used for multicast routing within a single administrative domain (AS).

One of the first suggestions for the realization of inter-domain multicast routing has been the combination of PIM-DM [36] and PIM-SM [44]. In this approach, PIM-DM is being used as the intra-domain routing protocol, and the source specific trees constructed and maintained by PIM-DM in each domain are connected in a shared tree constructed by PIM-SM. Due to a large control overhead caused by, among other things, advertising RPs, this scenario is not applicable to the Internet.

A longer-term solution that is currently being investigated uses a hierarchical addressing scheme called the Multicast Address-Set Claim (MASC) [83] protocol and the Border Gateway Multicast Protocol (BGMP) [97] that construct the bidirectional inter-domain shared trees, connecting individual multicast trees built in a domain.

The most commonly used inter-domain multicast routing solution to this end is the PIM-SM/MBGP/MSDP protocol suite. The Multicast Source Discovery Protocol (MSDP) [47] is a mechanism to connect multiple PIM Sparse-Mode (PIM-SM) domains together. It works in the following way: Each PIM-SM domain's RP communicates with an MSDP speaking RP in another domain via a TCP connection. In this way, a virtual topology consisting of MSDP peering RPs is created. These connections are used to exchange information about sources for particular groups in different domains. When a local RP receives traffic from an active source in its domain, it sends Source Active (SA) messages to its MSDP peers. Each SA message contains the IP address of the source, the multicast group address, and the IP address of the originating RP, i.e. the RP that is sending the SA. These messages are sent periodically, for as long as the source is active. Each MSDP peer, upon receiving the SA message, controls whether there are any receivers in its local domain interested in receiving the traffic from the advertised source. If so, that RP will send a PIM-SM join message directly back to the source (and not to the RP in the AS of the source), creating a branch of the forwarding tree towards the sender. In addition, each MSDP peer forwards the SA message it just received away from the originating peer RP, after performing a RPF check back to it. The RPF check is performed based on paths advertised by Multiprotocol BGP (MBGP). Multiprotocol BGP (MBGP) [13] is an extension to BGP, in the sense that is it able to distinguish between unicast and multicast topologies. The differences in these topologies occur because, for example, there may exist parts of the network that, due to policy regulations, refuse multicast traffic.

The advantage of MSDP is that it is easy to implement and that it solves the third-party dependencies problem-it allows each domain not to depend on a competitor's RP for transmitting the multicast traffic, but to get data directly from the multicast source, wherever it is located.

However, PIM-SM/MSDP is only an interim solution due to, firstly, scalability issues: if there would be thousands of multicast groups, the number of SA messages flooding the network would become too large to handle. Two other problems related to MSDP are join latency and bursty sources problems. Join latency problem arises because RPs with no receivers for the particular group discard the SA message. If a receiver would join before the new SA is issued, it would not receive the multicast content until the next SA message is issued. Bursty sources, as their name implies, send data in bursts, that can be separated by several minutes long time intervals. The fact that MSDP multicast routing state is established only after the information on the active sources is transmitted might cause some of the initial packets from a bursty source to be lost. This can occur because when a local RP receives a packet from a source, it sends an SA message to its peers, and, after establishing the reverse forwarding path, the receivers can join the source. However, it takes some time to both forward SA messages and to establish forwarding state in other RPs. Hence, by the time interested receivers would join the source, the initial packets may already have been dropped. If the silent periods between packets transmission is longer than the forwarding state timeout (set by default to three minutes), forwarding state will be removed. If the source would resume with sending packets only at that moment, then the whole process of establishing the routing state should start all over, with the initial data packets being lost again.

3.1.2 Multicast service models: Any Source Multicast (ASM), Source Specific Multicast (SSM) and Explicit Multicast

The traditional multicast model, as defined in RFC1112 [34], is also known under the name Any Source Multicast (ASM). Its name reflects the capability to allow one or many sources to generate a multicast group's traffic. The main design concept of this model is a concept of a host group, a group of users interested in a particular multicast application. A group is identified by a single class-D destination address. As already stated in Chapter 1, this model is an open model: any host can join and leave a given multicast group any time, and there are no restrictions on their location nor number. Any host can send a packet to that group address, and have it delivered to all members of the group. Moreover, the sender of the content does not need to be a member of the particular group. This model supports both source-specific and group-shared trees, and, accordingly, both one-to-many (e.g. audio/video distribution, push media and file distribution) and many-to-many (e.g. video conferencing, multiuser games, distance learning) type of applications.

To this end, all multicast-enabled networks have been designed to support the ASM service model. The open character of this architecture, however, has caused several important deployment problems. Firstly, finding a globally unique multicast addresses for a group is difficult. It is always possible that another multicast group uses the same multicast address. At this moment there is no technically feasible solution for preventing address collisions in ASM among multiple applications. Next, receivers in the traditional model are susceptible to flooding attacks, since they are not able to specify from which particular source(s) they want to receive traffic from.

Source Specific Multicast (SSM) [58, 15] is a multicast service model developed to solve those issues. SSM is a service model that supports multicast delivery from only one specified source to its receivers. In this model, a notion of "channel" is defined. Each channel consists of precisely one sender and arbitrarily many receivers. Each channel is identified by an (S,G) pair, where S is an address of a source and G is an SSM destination address. Receivers interested to receive packets from S need to subscribe to channel (S,G). Hence, when a receiver subscribes to an (S,G) channel, it receives data sent only by source S. An IP packet will then be transmitted from source S to address G along a source-specific tree. SSM defines thus channels on a per-source basis, i.e. channel (S1,G) is different from channel (S2,G). This prevents the problem of global allocation of SSM destination addresses, and enables each source to be responsible for resolving address collisions for the various channels. At the same time, when a sender picks a channel (S,G) to transmit on, it is automatically ensured that no other sender will be transmitting on the same channel. Finally, the SSM model simplifies intra- and inter-domain routing, since there is no need for shared trees and rendezvous points any more. The knowledge of the source and group pair is assumed to come from "out-of-band," for example a webpage. Since the Internet address of the source is given explicitly, there is no need to run MSDP.

Due to the weaknesses of the ASM model, many felt [10, 41] that the ASM model should be abandoned and replaced by the SSM permanently. The problem however is the lack of support of many-to-many applications in the SSM architecture. The proponents of the SSM argue (see [10, 41]) that in the short term this is not a serious concern, since the multicast applications are momentarily dominated by one-to-many applications (Figure 3.3 indicates that the average number of sources per group is below 5). The most appealing of multicast applications could be supported, while the amount of complexity required would be vastly reduced. Some other classes of multicast applications that are likely to emerge in the future are few-to-few (e.g. private chat rooms, whiteboards), few-to-many (e.g. video conferencing, distance learning) and many-to-many (e.g. large chat rooms, multi-user games). The first two classes can be easily handled using a few one-to-many source-based trees. The issue of many-to-many multicasting service on top of an SSM architecture is an open issue at this point. However, some feel [59] that even many-to-many applications can be handled with multiple one-to-many instead of shared trees, or alternatively by using an application layer relay mechanism [107].

Although SSM would be preferable in many cases, it has been argued [72] that SSM is not sufficiently widely available to completely replace ASM at this moment. The big three streaming players (Windows Media, Real, and QuickTime) all support ASM multicast, as do basically all operating systems (Windows, Mac OS, and Unix), and most routers (all Cisco routers, all Juniper routers, etc.). The challenge at this moment is to extend this support to include SSM (currently supported by Windows XP but not by many applications).

Finally, a third approach to providing multicast services is the recently proposed explicit multicast (XCAST) [17] for small multicast groups. The main idea behind the XCAST is to have packet headers that contain a list of all the (unicast) IP addresses of the multicast group members. Intermediate routers forward the XCAST packets along the unicast shortest paths and branch if needed. If branch is needed, the packet is replicated on the outgoing links and the list of IP addresses in the header is split properly.

3.2 Deployment Status

Multicasting over a large portion of the Internet has first taken place in March 1992 over the Multicast Backbone (MBone) [43, 8]. The MBone is an overlay network on top of the Internet, that was used to provide a multicast facility to the Internet. The MBone can be best viewed as a collection of "islands" that support multicasting within their domains. The multicast routing function was provided in each island by routers running a daemon process called *mrouted*, which received unicast-encapsulated multicast packets on an incoming interface and forwarded them subsequently to the appropriate set of outgoing interfaces. The *mrouted* daemons (in different islands) were connected to each other via point-to-point IP connections (called tunnels) over the Internet. In this manner, the *mrouted* daemons and the tunnels that connect them formed a virtual network on top of the Internet The original multicast routing protocol, DVMRP [103], was used to create multicast trees.

Since 1992, the MBone has grown tremendously. As more and more routers became capable of handling multicast packets, the MBone became integrated into the Internet itself. This has marked the beginning of the evolution of intra-domain multicast, to be followed by inter-domain solutions. Today, almost all network routers are equipped with multicast capability. And with the increasing number of network operators that implement IP multicast today, the MBone era, consisting of tunnelling and DVMRP, is long behind us.

Nevertheless, ten years after its initial deployment, IP multicast is still not widely deployed. Recent data shows (see [3]) that around 5% of the routable prefixes in BGP are reachable by multicast, and that around 3.5% of active AS numbers are multicast enabled. Most of the larger Internet Service Providers (ISPs) provide IP multicast, including Sprint, Worldcom/UUnet, Multicast Technologies, Verio, IP-only.net, LavaNet, Spirit One, Naino, and several others. Research networks (such as SURFnet and Internet2), provide IP multicast as well. The major obstacle to a wider deployment of multicast are (smaller) edge ISPs, that, for reasons to be explained below, are reluctant to support it.

Figure 3.1 to Figure 3.3 (source [3]) illustrate the penetration of multicast into the Internet. One basic measure of the size of the multicast enabled Internet is the

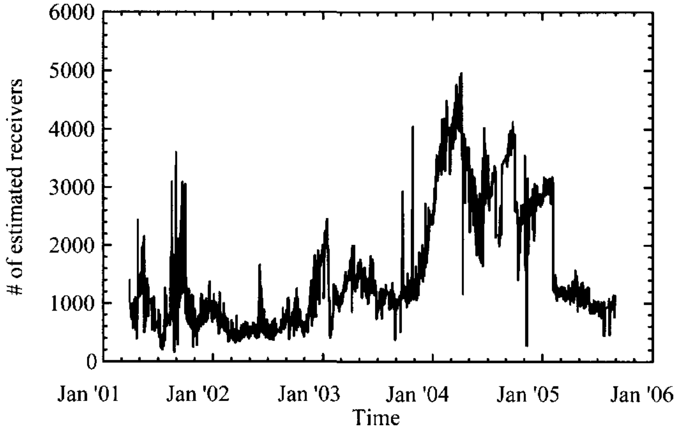


Figure 3.1: The relative size of the multicast enabled Internet.

number of address blocks (or prefixes) that have multicast routing. This number can be expected to grow proportionally to the number of sites that are multicast enabled, as each additional address block will tend to represent a new computer network that has become multicast enabled. Another measure for the growth of multicast enabled Internet is the total number of Autonomous Systems that enable multicast routing. Although multicast-equipped networks may vary widely in size, each new Autonomous System with multicast routing represents a new network enabled for multicast.

Figure 3.1 provides two independent estimates for the growth and the relative size of the multicast enabled Internet. The upper line represents the ratio of the number of address blocks (or prefixes) that have multicast routing to the number of address blocks (or prefixes) that have unicast BGP routing (as seen from the Multicast Technologies Autonomous System [3]). The lower line shows the ratio of the number of multicast enabled Autonomous Systems to the total number of Autonomous Systems with BGP routing. Both measures are growing, although the address block ratio tends to be higher. A reasonable estimate is that the actual penetration of multicast into the Internet lies somewhere between these two lines.

Figure 3.2 gives an indication on the usage of multicast in the Internet, as it shows the number of Autonomous Systems with multicast routing, together with the number of Autonomous Systems with active multicast groups. Finally, Figure 3.3 shows the ratio of the number of multicast sources and the total number of groups. Hence, it represents an estimate of the number of sources for each ASM group.

There are several technical and non-technical issues that have stalled multicast wide-

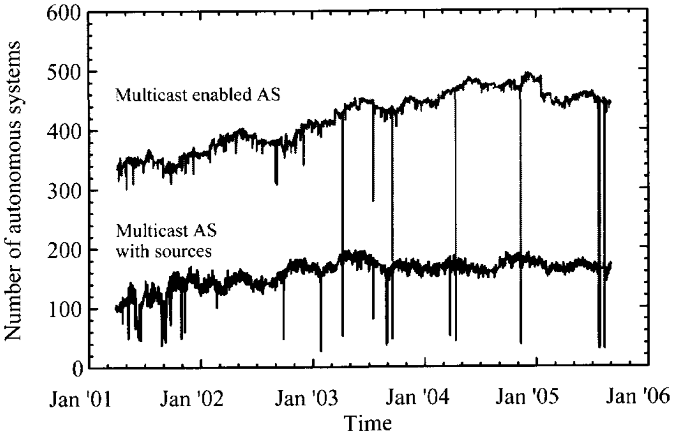


Figure 3.2: Autonomous Systems in the multicast-enabled Internet: total and those with active sources.

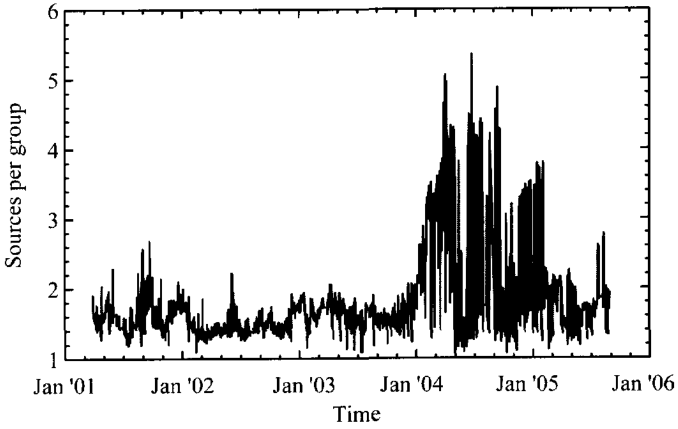


Figure 3.3: The average number of ASM sources per group.

spread deployment. Some of these issues have been alleviated in the years of research, however, other remain unresolved. In the sequel we will describe and discuss some of them.

3.2.1 Deployment Challenges

Group Management -Group management can be defined as a set of access control functions that determine who may send to and receive from a particular multicast group. The traditional ASM service model does not consider group management, including receiver authorization, transmission authorization, and group creation. This can induce several problems, such as spamming (a high-rate useless data is transmitted to the multicast group, causing congestion and packet loss) drowning out the authentic sources (transmitting the alternate data and changing the content of the session) and unauthorized reception of multicast data (including pay-for content, such as pay-per-view events). These problems affect both the ISPs and the content providers, as well as multicast users themselves.

Address Allocation- One of the problems in ASM that will occur as multicast becomes more popular is a multicast address allocation. Multicast address allocation refers to assigning to each application a unique address from a globally-shared address space. In the current IP protocol version, IPv4, the multicast address space consists of $s = 2^{28}$ distinct addresses. Since the address space is limited, addresses must be re-used over time. The current multicast address space is unregulated, and therefore nothing prevents applications from sending data to any multicast address. Members of two sessions will receive each other's data if separate addresses are not chosen. A lack of address allocation mechanisms poses a threat to ISPs, since they will be dealing with angry customers and carrying unwanted data. In addition, since packets from other sessions must be processed and dropped, address collision induces inefficiencies for multicast receivers and can lead to application inconsistencies.

Traditionally, multicast addresses are assigned to groups randomly, assuming that the collision probability is low. At this moment, the probability of an address collision is limited, but only because multicast has not yet become a popular interdomain service. The average router today has memory available for 1024 to 16384 (source address, group address) entries (default is set to 4096). The limited memory of routers in the current deployed Internet limits the probability of address collision, because new groups cannot be created after memory runs out. The probability of at least one collision in a set of X multicast addresses is equal to

$$Pr[collision] = 1 - \frac{s\,(s-1)\,(s-2)\ldots(s-X+1)}{s^X}$$

(or $1 - e^{\frac{X(X-1)}{4s}}, X \ll s$). For a memory that can store 1024 addresses ($X = 1024$), the probability of collision is limited to 0.18 percent. However, as multicast gains popularity (it is conceivable that many thousands of multicast groups will be simultaneously in use in a large network) and routers reserve more memory for multicast addresses, the collision probability will become unacceptably high: For routers with memory that can store only 8192 addresses, the probability of a collision increases to about 12 percent.

Network Management- Network management refers to the debugging of problems that occur with the multicast tree during transmission, and the monitoring of utilization and operation patterns for the purpose of network planning. While intradomain multicast is relatively easy to deploy, providing interdomain multicast is complicated. Problems may occur when RPs and their associated sources are located in different domains. Dependence on an RP in the domain of another ISP makes it difficult to diagnose and debug problems. Most of the debugging tools so far are academic prototypes; none of them is robust enough to support commercial deployment. They only partially address the various issues in monitoring and debugging, and cannot identify all problems related to the current protocol architecture.

The three above-listed problems (group management, address allocation and network management) have been somewhat alleviated with the introduction of the SSM model (and IGMPv3 that supports it). First, the SSM model and IGMPv3 provide both source pruning for specific multicast groups, as well as source-specific joins. This makes spamming an SSM channel significantly more difficult than an ASM host group.

Furthermore, per-source (or channel) allocation as implemented in the SSM model eliminates the addressing problem as well, as the source address makes each channel unique. Other proposed solutions to addressing problem include static allocation and assignment [74], where the address space is divided equally between the autonomous systems in the Internet, and IPv6 addressing.

Finally, interdomain multicast and network management are simplified with SSM, since the Internet address of the source is given explicitly, and there is no need to run MSDP.

Multicast Security- Providing security for multicast-based communication is inherently more complicated than for unicast-based communication because multiple entities participate, most of which will have no trusted relationships with each other. Future multicast security should provide four distinct mechanisms: authentication, authorization, encryption and data integrity. Authentication is the process of forcing hosts to prove their identities so that they may be authorized to create, send data to, or receive data from a group. Authorization is the process of allowing authenticated hosts to perform specific tasks. Encryption would ensure that eavesdroppers cannot read data on the network. Data integrity mechanisms would ensure that the datagram has not been altered in transit.

Lack of a proper business model: One of the most significant reasons behind the slow deployment of multicast has been the lack of a good business scenario for the providers. In terms of deployment and management, providing multicast is more costly than unicast. Therefore, ISPs are only motivated to provide multicast if the savings in bandwidth would be higher than the deployment and management costs. This threshold is referred to as multicast deployment's sweet spot: the point where the gained benefit outweighs the additional costs. The cost of a network service can be defined as the sum of network related costs, infrastructure costs and management costs (in terms of human resources). Multicast does have a high initial cost, higher than unicast. However, the events that attract a large audience (millions of interested users), have the potential to overwhelm any (large) collection of servers and available network bandwidth. This makes multicast not only a profitable and useful service for both content and the network providers, but also a necessary one. As a number of users grows, so does the benefit of multicast.

Chapter 4

Properties of Multicast Routing Trees

In the previous chapter, we have discussed some of the issues that stalled multicast deployment, and concluded that one of the leading issues is a lack of a business incentive for network providers. In order to provide a trustworthy model for multicast business scenarios, it is necessary to establish a proper model for multicast trees, valid for any type of network topology. In this chapter, we look into the properties of multicast routing trees. We believe that understanding and quantifying these properties would lead us to the establishment of a realistic model for multicast trees and, subsequently, to a reliable business model. This chapter further discusses possible implications of our analytical derivations for the network providers. In Chapter 5, our theoretical findings will be validated via Internet measurements.

4.1 The gain of multicast over unicast

One of the main benefits of multicast is that it economizes on the number of link-traversals: only one copy of the data packet traverses each link along the shortest path to destinations. Hence, multicast reduces the overall network load. In this section, we focus on the efficiency, or gain, of multicast in terms of network resource consumption compared to unicast. Specifically, we concentrate on a one-to-many type of communication, where a source distributes messages (packets) along the shortest path to m different, uniformly distributed destinations.

4.1.1 Gain: Chuang-Sirbu law and log-log scale insensitivity

Quantifying the network savings of multicast has been initiated by Chuang and Sirbu [32], and further investigated by Philips *et al.* [81] and Chalmers and Almeroth [26]. In their study presented in [32], Chuang and Sirbu made two modeling assumptions. First, they assumed that multicast packets are being delivered along a Shortest Path Tree from a source to m destinations. As most of the current multicast protocols forward packets based on the (reverse) shortest path, the assumption of shortest-path tree delivery is rather realistic. Secondly, they assumed that m destinations are uniformly chosen out of N nodes. The uniformity assumption has further been validated in [81] by Philips *et al.* They concluded that, for large m and N, deviations from the uniformity assumption are negligibly small. Internet measurements reported in [81] and [26] seem to support the assumption as well.

The gain (savings) of multicast compared to unicast can analytically be expressed in the following way. Let us denote by $H_N(m)$ the number of link traversals, or hops, in the Shortest Path Tree rooted at a particular source to m randomly chosen destinations. In unicast, messages have to be sent m times from the source to each destination. Hence, unicast uses on average $f_N(m) = mE[H_N]$ link-traversals (hops), where $E[H_N] = E[H_N(1)]$ is the average number of hops of a message to a uniform location in the graph containing N nodes. If we define for multicast $g_N(m) = E[H_N(m)]$ to be the average number of hops in the Shortest Path Tree rooted at a source to m randomly chosen distinct destinations, then, of course, $g_N(m) \leq f_N(m)$. For the extreme sizes of the multicast group, we have simple expressions: $g_N(1) = f_N(1) = E[H_N]$ while $g_N(N-1) = N-1$ reflecting the number of links in a (complete) spanning tree.

Chuang and Sirbu defined the normalized cost of a multicast tree, as the ratio $E[H_N(m)]/E[H_N]$. Empirically, via simulations on various network topologies, they obtained results which when plotted on log-log scale suggested a power-law relationship

$$E[H_N(m)]/E[H_N] \approx m^{\nu} \tag{4.1}$$

where $E[H_N(m)]$ is the total number of multicast links in the tree, $E[H_N]$ is the average number of hops between any two nodes in the network, ν is the economy-of-scale factor taking the value of 0.82 and m is the number of multicast receivers.

The Chuang-Sirbu law is the result most referred to when discussing the gain of multicast. Yet, Van Mieghem *et al.* [101] have proven that the Chuang-Sirbu law cannot hold for all m. The problem with Chuang-Sirbu law is that it has been deduced from an observed straight line in a log-log plot. Many functions, however, can be expanded in some interval as a polynomial, hence, one must be particularly cautious when drawing conclusions based on an insensitive log-log scale plot.

Below, we merely list some of the more important results obtained on the gain (efficiency) of multicast in [101].

4.1.2 Gain: Theory

The average number of hops $E\left[H_N\left(m\right)\right]$ in the Shortest Path Tree rooted at a particular source to m randomly chosen destinations, in the graphs of class $G_p\left(N\right)$, with independently chosen links with probability p and with uniformly (or exponentially) distributed link metrics w, has been shown in [101] to obey

$$E\left[H_N\left(m\right)\right] = mN\left(\frac{\psi(N)-\psi(m)}{N-m}\right)-1, \tag{4.2}$$

where $\psi(x)$ is the digamma function. For large N, we have the accurate asymptotic,

$$E\left[H_N\left(m\right)\right] \sim \frac{mN}{N-m}\log\left(\frac{N}{m}\right)-\frac{1}{2} \tag{4.3}$$

Unicast uses on average $f_N(m) = mE\left[H_N\right]$ hops to deliver the message to m randomly chosen destinations. Since in the RGU class of graphs and for large N $E\left[H_N\right]$ is given with (2.4), $f_N(m)$ follows

$$f_N(m) \sim m(\log N + \gamma - 1), \tag{4.4}$$

This scaling law clarifies the empirically derived Chuang-Sirbu law in $G_p(N)$: for small values of m, when plotted on the log-log scale, functions $\left(\log N + \gamma - 1\right) m^{0.8}$ and $\frac{mN}{N-m}\log\left(\frac{N}{m}\right)-\frac{1}{2}$ look very much alike. Van Mieghem *et al.* [101] have further shown that for the complete graph with exponentially distributed weights, and for m small with respect to N, the ratio $E\left[H_N\left(m\right)\right]/mE\left[H_N\right]$ increases about linearly with m on a log-log scale, explaining the empirical Chuang-Sirbu law.

To conclude, although seemingly close to power-law, $E\left[H_N\left(m\right)\right] = g_N\left(m\right)$ does not follow a power-law for all m.

We will use the above results in our mathematical analysis of the stability of multicast trees in the following section, as well as in our discussion on possible implications of multicasting for network providers in Section 4.4.

4.2 Stability of Multicast Routing Trees

Apart from the dynamics of topology updates, IP multicast design offers the possibility of members joining and leaving a group at any time. This activity requires the multicast tree to be dynamically updated (e.g. branches without multicast members must be discarded). These changes imply that the forwarding of IP multicast packets may change dramatically, resulting in undesirable transient routing effects.

The goal of this section is to investigate and quantify multicast stability, in particular, to determine the probability density function for the number of branches that change if one user joins or leaves the group. In addition, we quantify the common belief (see e.g. the book of Huitema [60]) that Steiner Trees are more instable than Shortest Path Trees. This intuitive supposition is the other significant reason for not deploying Steiner Trees in multicast routing protocols (as stated in [60]), in addition to their increased computational complexity. Instead, most of the current multicast protocols forward packets based on the (Reverse) Shortest Path. The SPT algorithm does not necessarily result in a tree that economizes on network resources but it is easy to compute and it offers minimum delay.

This section is organized as follows. In Section 4.2.1, the stability of the multicast tree is defined and basic theoretical results are deduced. Section 4.2.2 presents simulation results for both the Shortest Path Tree (SPT) and the Steiner Minimum Tree (SMT). Shortest Path Trees and Steiner Minimum Trees are compared in Section 4.2.3.

4.2.1 Stability: Theory

Inspired by Poisson arrival processes, at a single instant of time, we assume that either no or one group member can leave. In the sequel, we do not make any further assumption about the time-dependent process of leaving/joining a multicast group and refrain from dependencies on time. As a measure for the stability of the multicast tree, the number of links in the tree that change after one multicast group member leaves the group has been chosen. If we denote this quantity by $\Delta_N(m)$, then, by definition of $g_N(m)$, the average number of changes equals

$$E\left[\Delta_N(m)\right] = g_N(m) - g_N(m-1) \tag{4.5}$$

Since $g_N(m)$ is concave (see [101, Theorem 2]), $E\left[\Delta_N(m)\right]$ is always positive and decreasing in m. If the value of m is extended to real numbers, then $E\left[\Delta_N(m)\right] \approx g'_N(m)$, which simplifies further estimates.

The situation where on average less than 1 link changes if one multicast group member leaves may be regarded as a stable regime. Since $E\left[\Delta_N(m)\right]$ is always positive and decreasing in m, this regime is reached when the number of the receivers m exceeds m_1, which satisfies $E\left[\Delta_N(m_1)\right] = 1$. For example, for the recursive tree, which is the Shortest Path Tree (as shown in [100]) for the class RGU[1], this condition approximately follows from (4.3) as

$$E\left[\Delta_N(m)\right] \sim \frac{mN}{N-m}\log\left(\frac{N}{m}\right) - \frac{(m-1)N}{N-m+1}\log\left(\frac{N}{m-1}\right) \tag{4.6}$$

Let $y = \frac{m}{N}$, then $0 < y < 1$ and

$$\frac{E\left[\Delta_N(m)\right]}{N} \sim \frac{-y}{1-y}\log y + \frac{(y-1/N)}{1-(y-1/N)}\log\left(y - \frac{1}{N}\right)$$

---

[1] For the definition of the class RGU, see Chapter 2.

After expanding the second term in a Taylor series around y to first order in $\frac{1}{N}$,

$$E\left[\Delta_N(xN)\right] \sim \frac{y-1-\log y}{(1-y)^2} + O\left(\frac{1}{N}\right)$$

Hence, for large N, $E\left[\Delta_N(y_1 N)\right] \sim 1$ occurs when $y_1 = 0.3161$, which is the solution in y of $\frac{y-1-\log y}{(1-y)^2} = 1$. For the class RGU, a stable tree as defined above is obtained when the multicast number of receivers m is larger than $m_1 = 0.3161N \approx \frac{N}{3}$. In the sequel, since m_1 is high and of less practical interest, we will focus on multicast group sizes smaller than m_1. The computation of m_1 for other graph types turns out to be difficult. Since, as we will demonstrate in Chapter 5, the comparison with Internet measurement shows that formula (4.3) provides a fairly good estimate, we expect that $m_1 \approx \frac{N}{3}$ also approximates the stable regime in Internet well.

The following theorem quantifies the stability in the class RGU.

Theorem 1 *For sufficiently large N and fixed m, the number of changed edges $\Delta_N(m)$ in a random graph $G_p(N)$ with uniformly distributed link weights tends to a Poisson distribution,*

$$\Pr\left[\Delta_N(m) = k\right] \sim e^{-E[\Delta_N(m)]} \frac{\left(E\left[\Delta_N(m)\right]\right)^k}{k!} \tag{4.7}$$

where $E\left[\Delta_N(m)\right] = g_N(m) - g_N(m-1)$ and $g_N(m)$ is given by (4.2) or approximated by (4.3).

Proof. In Chapter 2 we have stated that it has been shown ([100][101]) that the Shortest Path Tree from a source to an arbitrary node in the random graph $G_p(N)$ with uniformly (or exponentially) distributed link weights, is a Uniform Recursive Tree for large N. In addition, the number of hops (the hopcount H_N) from that source to an arbitrary node tends, for large N, to a Poisson random variable with mean $E\left[H_N\right] \sim \log N + \gamma - 1$, where γ is Euler's constant ($\gamma = 0.5772156\ldots$). Hence, $\Delta_N(m)$ is the random variable that counts the absolute value of the difference between the hopcount $H_N(m)$ from the source to user m and the hopcount $H_N(m-1)$ from the source to the user closest in the tree to m, which we label by $m-1$. Both users m and $m-1$ are not independent, nor are the two random variables $H_N(m)$ and $H_N(m-1)$ independent in general, due to a possible overlap in their paths. If the shortest paths from the root to each of the two users m and $m-1$ overlap, there always exists a node in the Shortest Path Tree, say node B as illustrated in Figure 4.1, that sees the partial shortest paths from itself to m and $m-1$ as non-overlapping and independent. Since the Shortest Path Tree is a Uniform Recursive Tree, the subtree rooted at that node B (shown as a dotted line in Figure 4.1) is again a Uniform Recursive Tree[2]. With respect

---

[2] Recall that a Uniform Recursive Tree possesses the property that any new node N has equal probability to be attached to any of the $N-1$ nodes already in the tree.

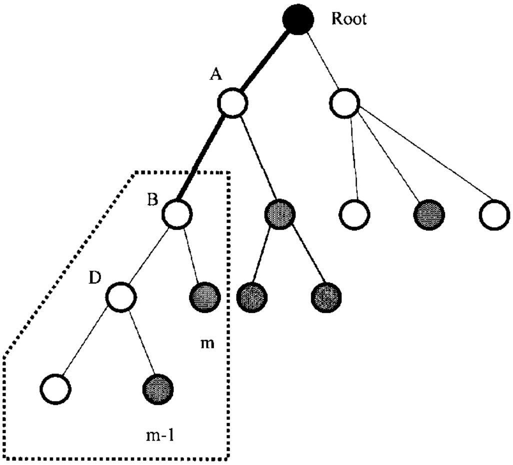


Figure 4.1: A sketch of a uniform recursive tree, where $H_N(m) = 3$ and $H_N(m-1) = 4$ and the number of links in common is two (shown in bold Root-A-B).

to B, the nodes m and $m-1$ are uniformly chosen. We denote the unknown number of nodes in that subtree rooted at B by $\nu(m) \leq N$. We have that $\nu(m) \leq \nu(m-1)$ because by adding a group member, the size of the subtree can only decrease. For large N and small m, $\nu(m)$ is large such that the above mentioned asymptotic law of the hopcount applies. If both m and N are large, $\nu(m)$ will become too small for the asymptotic law to apply (a fact illustrated by the simulations in Section 4.2.2). Thus, for fixed m and large N, this implies that $\Delta_N(m)$ tends to Poisson random variables with mean $E[\Delta_N(m)]$. For any graph and any m and N, relation (4.5) applies. Since $E[\Delta_N(m)]$ can be explicitly computed as (4.6), this completes the proof. ■

Remark that the proof can be extended to a general topology. Assume for a certain class of graphs that the pdf of the hopcount $\Pr[H_N = k]$ and the multicast efficiency $g_N(m)$ can be computed for all sizes N. The subtree rooted at B is again a Shortest Path Tree in a subcluster of size $\nu(m)$, which is an unknown random variable. An argument similar as the one in the proof above shows that

$$\Pr[\Delta_N(m) = k] = \Pr\left[H_{\nu(m)} = k\right]$$

This argument implicitly assumes that all multicast users are uniformly distributed over the graph. By the law of total probability,

$$\begin{aligned}\Pr\left[H_{\nu(m)} = k\right] &= \sum_{n=1}^{N} \Pr\left[H_{\nu(m)} = k | \nu(m) = n\right] \Pr\left[\nu(m) = n\right] \\ &= \sum_{n=1}^{N} \Pr\left[H_n = k\right] \Pr\left[\nu(m) = n\right]\end{aligned}$$

which, unfortunately shows that the pdf of $\nu(m)$ is required to specify $\Pr\left[\Delta_N(m) = k\right]$. However, we can proceed further in an approximate way by replacing the unknown random variable $\nu(m)$ by its best estimate, $E\left[\nu(m)\right]$. In that approximation, the average size $E\left[\nu(m)\right]$ of the shortest path subtree rooted at B can be specified, at least in principle, with the use of (4.5). Indeed, since $E\left[H_{E[\nu(m)]}\right] = \sum_{k=1}^{E[\nu(m)]-1} k \Pr\left[H_{E[\nu(m)]} = k\right]$, by equating

$$E\left[H_{E[\nu(m)]}\right] = g_N(m) - g_N(m-1)$$

a relation in one unknown $E\left[\nu(m)\right]$ is found and can be solved for $E\left[\nu(m)\right]$. In conclusion, we end up with the approximation

$$\Pr\left[\Delta_N(m) = k\right] \approx \Pr\left[H_{E[\nu(m)]} = k\right]$$

which roughly demonstrates that, in general, $\Pr\left[\Delta_N(m) = k\right]$ is likely related to the hopcount distribution in that given class of graphs.

Unfortunately, for very few types of graphs, both the pdf $\Pr\left[H_N = k\right]$ and the multicast gain $g_N(m)$ can be computed. This fact augments the value of Theorem 1, although the class RGU is not a good model for the graph of the Internet. Fortunately, the Shortest Path Tree deduced from that class seems a reasonable approximation (as shown in [100]) and sufficient to provide first order estimates. Moreover, its relatively simple analytic character is desirable in modeling problems.

4.2.2 Stability: Simulation Results

The main goal of the simulations is to verify the quality of the asymptotic result in Theorem 1. This section is devoted to that purpose. In addition, results for the Steiner Minimum Tree on the same type of graphs for the class RGU are presented and compared to those of the corresponding Shortest Path Tree.

In order to anticipate frequently received criticism about the class RGU, the value of the results only applies to this class RGU and no attempt is made to correlate these results to the current Internet, although the previous section did so. The main reasons are as follows:

1. The topology of the Internet is currently not sufficiently known to categorize the Internet as a type or an instance of a class of graphs. The Internet is most likely

best seen as an organism changing over time; there does not exist a fixed Internet topology and, hence, a class specification is desirable, in particular for simulations. There are measurements (on a part) of the Internet that show that the Internet graph is sparse (low link density p). The measurements further indicate that the distribution of the degrees (number of links per node/router) is likely polynomially distributed (as mentioned in Chapter 2 and as will be discussed in more detail in Chapter 5) with exponent close to -2.2 (see. e.g. [46]). Unfortunately, these measurements only reveal a part of what we need to know.

2. For any routing problem, in addition to the network topology, we also need knowledge of the link weight distribution. Older topology models and generators were more likely to define all links with unit weight ($w = 1$). However, as briefly discussed in Chapter 2, it makes sense to distinguish between a satellite link, a large bandwidth link and a smaller, or legacy link. Hence, not all link weights will be equal to $w = 1$.

Even if more realistic topology generators (such as e.g. gt-itm [108]) are used, the second problem of the link weight distributions remains debatable. The link weight distribution is equally important as the topology of the graph itself. It has been shown in [100] that for N large enough (in practice $N > 50$), the dependency of the hopcount of the shortest path on the link density p (i.e. the number of links in the graph) becomes insignificantly small. Moreover, by attaching a certain weight to a link, the specific details of the underlying topology may be shielded (or become irrelevant) in a routing problem [75].

Shortest Path Tree (SPT)

We confine ourselves to graphs of the class RGU with $N \geq 100$ and with link density $p = 0.2$. The value of p is arbitrarily chosen since, as stated above, the hopcount of the shortest path is insensitive to the value of p in sufficiently large graphs. For each graph of N nodes, we define the number of multicast receivers in the network, and the source node. For each N and p, 10^5 topologies are generated randomly. The connectivity is tested using Prim's minimum spanning tree algorithm [33]. Only if the generated topology is connected, m nodes out of $N-1$ (the node number one was defined as a source node) are uniformly chosen, and the Shortest Path Tree is computed using a modification of Dijkstra's algorithm. The Dijkstra algorithm is modified in such a way that the algorithm stops after finding the paths to all m destinations, as explained in Section 2.1. The number of edges in the tree was computed as well as the number of edges in the tree that interconnects one (uniformly chosen) multicast user less. The difference of those two values was stored in a histogram, from which the probability density function was deduced, as well as the mean $E[\Delta_N]$ and the variance $var[\Delta_N]$ of the number of changed edges. These two variables ($E[\Delta_N]$ and $var[\Delta_N]$) are plotted

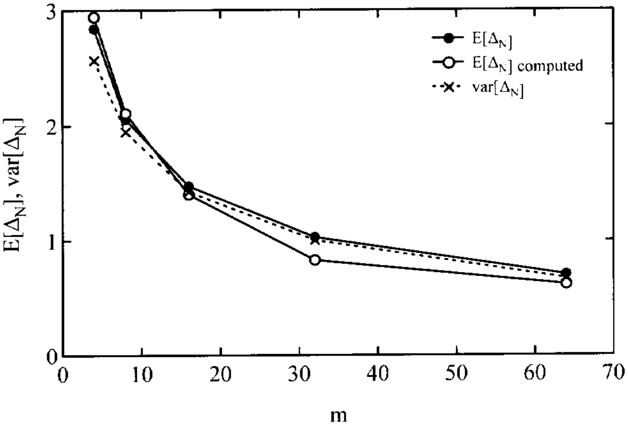


Figure 4.2: SPT: The mean and variance of Δ_{100}.

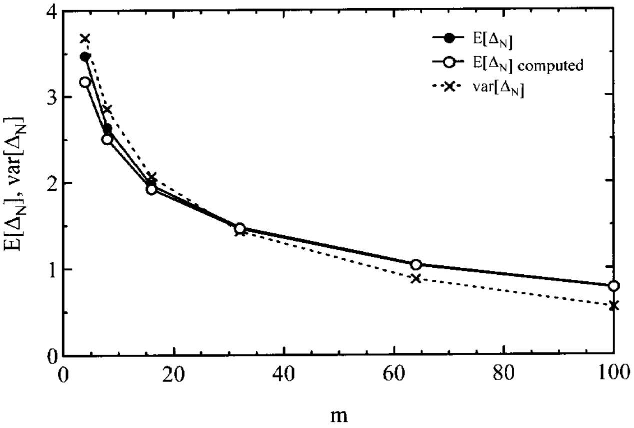


Figure 4.3: SPT: The mean and the variance of Δ_{200}.

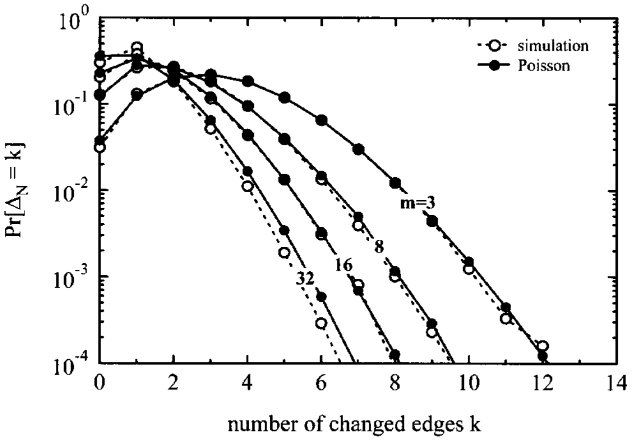


Figure 4.4: SPT: Pdf $\Pr[\Delta_N = k]$ for $N = 100$ and $m < N/3$.

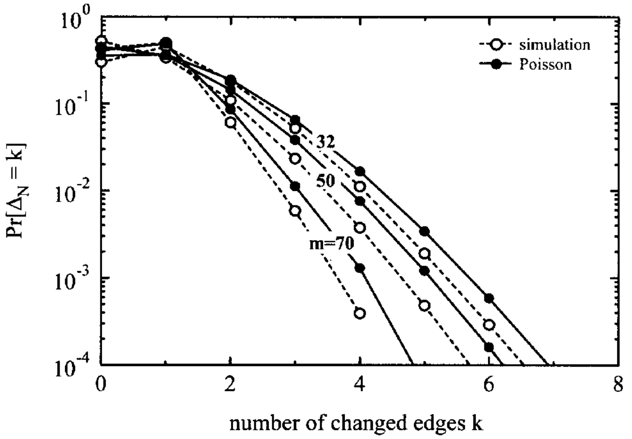


Figure 4.5: SPT: Pdf $\Pr[\Delta_N = k]$ for $N = 100$ and $m > N/3$.

as a function of the number of receivers m, for two different values of N (100 and 200 respectively) in Figures 4.2 and 4.3. However, for larger values of N, this simulation process is time consuming, and not efficient. Therefore, for N larger than 500, we used a Markov discovery process to find the shortest paths from the source node to the other multicast group members. The Markov discovery process has been explained in detail in [100]. The Markov discovery process allows us to compute the Shortest Path Tree very efficiently in large graphs (even up to 10^5 nodes) of the class RGU.

We observe that the mean $E[\Delta_N]$ determined via the simulations, and the mean $E[\Delta_N]$ computed by (4.6) are almost identical. Another important observation is that there is an area where the mean $E[\Delta_N]$ and the variance $var[\Delta_N]$ tend to each other. Since this is a property of the well-known Poisson distribution, we are led to the conclusion that the probability density function of the number of changed edges Δ_N is very likely a Poisson distribution. In Figures 4.4 to 4.7, simulation results together with the Poisson law (4.7) are plotted in the dotted and the solid line respectively, as a function of the number of changed edges, with the number of receivers m as a parameter.

Figures 4.4 to 4.7 show that for $m < \frac{N}{3}$ (equivalent to $E[\Delta_N] > 1$), the probability density function is remarkably well described by the Poisson distribution. For $m > \frac{N}{3}$, the noticeable differences between the mean $E[\Delta_N]$ and the variance $var[\Delta_N]$ appear, and there are significant deviations of the probability density function from the Poisson distribution. The explanation is that the size $\nu(m)$ of the subtree rooted at B as illustrated in Figure 4.1, becomes too small to justify a Poisson law for the hopcount in that subtree. But, as we have already explained in Section 4.2.1, if the average number of changed links is less than one, the multicast tree can be considered as stable.

Figure 4.8 and 4.9 represent results obtained from the Markov discovery process, for $N = 1000$. These figures show that, for $N = 1000$, the probability density function matches the Poisson distribution (4.7) even for larger values of m.

Finally, the effect of the link weight distribution on the number of changed branches Δ_N in the Shortest Path Tree is illustrated in Figure 4.10. For graphs of the class $G_p(N)$, this Figure 4.10 compares the pdf $\Pr[\Delta_N = k]$ obtained with uniformly (or exponentially) distributed and with constant ($w = 1$) link weights. Earlier in [100], it was shown that, for all link weights equal in $G_p(N)$, the probability that the hopcount exceeds 2 hops precisely equals

$$\Pr[H_N > 2] = (1-p)\left[1-p^2\right]^{N-2}$$

and very rapidly decreases with N for all link densities $p > \frac{1}{\sqrt{N}}$. This phenomenon is also observed in the behavior of Δ_N in Figure 4.10 and supports the generalization of the Poisson law (4.7) - which is deduced for uniformly (or exponentially) distributed link weights - that $\Pr[\Delta_N(m) = k]$ is reasonably well approximated by $\Pr\left[H_{E[\nu(m)]} = k\right]$.

Figure 4.10 also seems to indicate that less variability in the link weight distribution amounts to a higher stability of the shortest path multicast tree. Although concluded

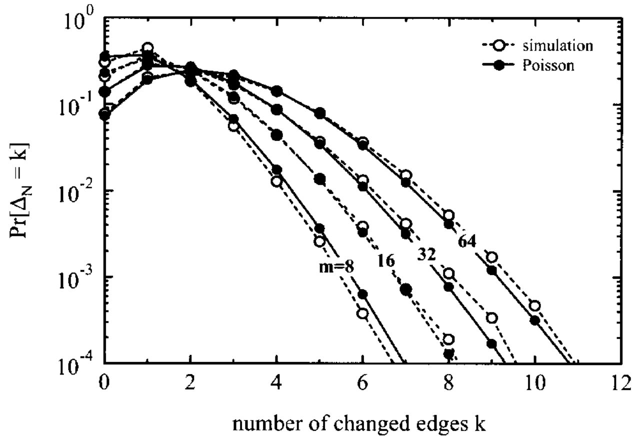


Figure 4.6: SPT: Pdf $\Pr[\Delta_N = k]$ for $N = 200$ and $m < N/3$.

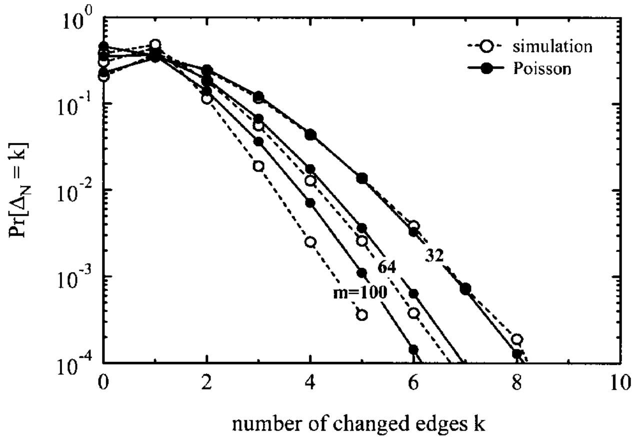


Figure 4.7: SPT: Pdf $\Pr[\Delta_N = k]$ for $N = 200$ and $m > N/3$.

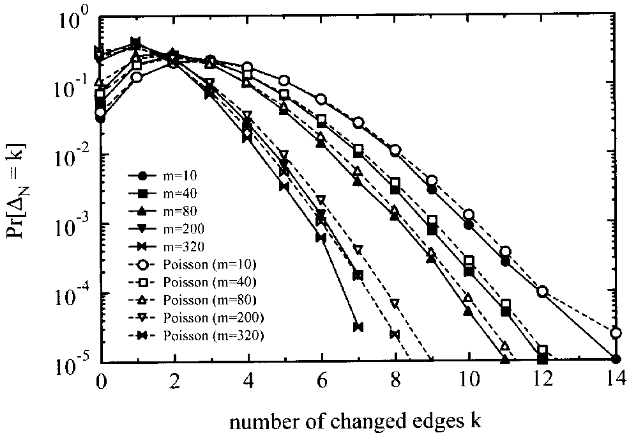


Figure 4.8: SPT: Pdf $\Pr[\Delta_N = k]$ for $N = 1000$ and $m < N/3$.

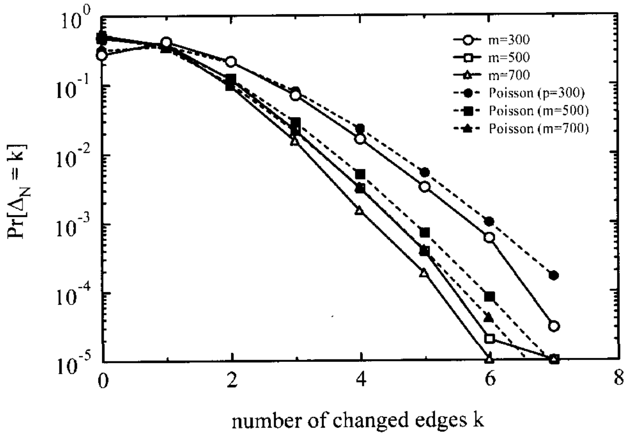


Figure 4.9: SPT: Pdf $\Pr[\Delta_N = k]$ for $N = 1000$ and $m > N/3$.

from the class $G_p(N)$, similar simulations with more realistic topologies generated by gt-itm [108] confirm this stable Shortest Path Tree behavior[3] (Figure 4.11).

In conclusion, the simulation results indicate that, in spite of the applicability of Theorem 1 to an asymptotic regime (large N and fixed m), the law (4.7) seems to have a wider validity region. This feature has previously been observed in [102] for the probability distribution of the hopcount between two arbitrary chosen nodes. In [102] it has been theoretically demonstrated that the asymptotic law given with (2.6) possesses the property of almost sure behavior. This property implies that each hopcount distribution obtained by measurements from a source to a certain number of destinations will closely resemble (2.6). A wider validity region of Theorem 1 suggests the robustness of the Poisson law against certain changes in the modeling assumptions which might be associated with almost sure behavior.

Steiner Minimum Tree

We continue by presenting corresponding results obtained for the Steiner Minimum Trees. The simulation process is similar to the one used for generating the Shortest Path Tree. Again we performed simulations in the class RGU. We generated 10^5 random graphs of that class RGU. In each graph, $m_t = m + 1$ multicast group members are chosen uniformly out of the N possible nodes. Depending on m_t, the Steiner Minimum Tree [45] is generated using different algorithms. For $m_t = 2$, the Steiner Minimum Tree (SMT) problem reduces to the computation of the shortest path between those two users. If $m_t = N$, the MST is actually the (complete) minimum spanning tree, and is computed with Prim's algorithm. For $2 < m_t < N$, the SMT problem belongs to the class of NP-complete problems. Certain reductions in the topology [45] decrease the number of nodes and links to a reduced graph, and increase the speed of simulations. In spite of the implemented reductions, the simulation process is extremely time consuming for large N. Therefore, we confine ourselves to graphs where N is not larger than 20 [4]. In each graph, the SMT is computed for $m_t = m + 1$ and m members of the multicast group. The difference Δ_N in the number of the links forming these trees was stored in a histogram, from which the probability density function was deduced.

Influence of the link weight distribution. For the class of $G_p(N)$ with various polynomial link weight distributions specified by the power exponent α,

$$\Pr\left[w \leq x\right] = x^{\alpha} 1_{0 \leq x \leq 1} + 1_{x \geq 1}$$

---

[3] 100000 iterations of transit-stub graph with $N = 100$ nodes have been performed.

[4] The computational time for a set of simulations of Steiner Minimum Trees on a SUN Solaris OS 5.8 workstation (CPU frequency of 360 MHz), for $N = 20$, in 2001, amounted to approximately 700 hours of simulations.

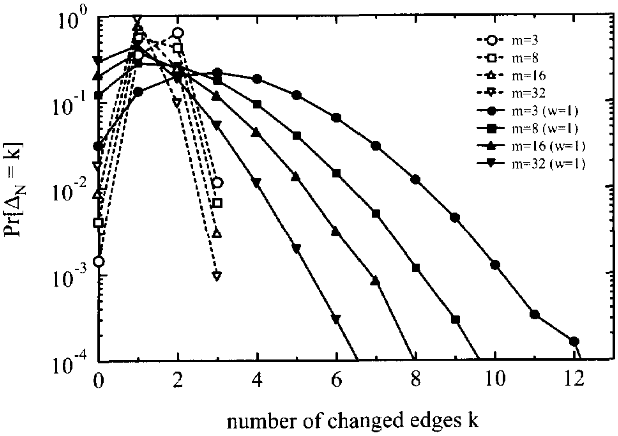


Figure 4.10: SPT: Comparison of the pdf $\Pr[\Delta_{100} = k]$ in the class $G_{0.2}(100)$ with uniformly distributed link weights (dotted line and transparent markers) and all link weights $w = 1$ (full line and full markers).

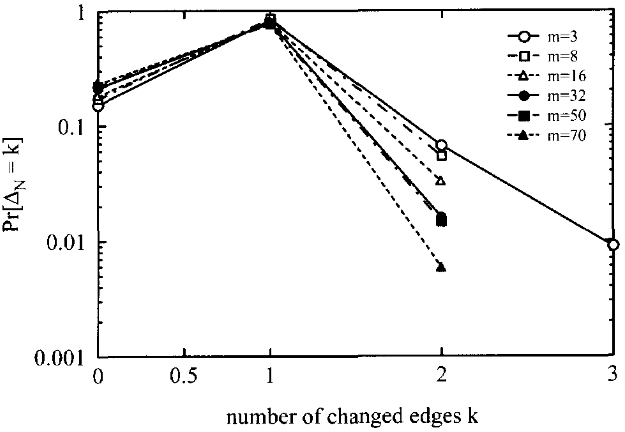


Figure 4.11: SPT: Pdf $\Pr[\Delta_N = k]$ on the topology generated with gt-itm generator (transit stub graph with 100 nodes).

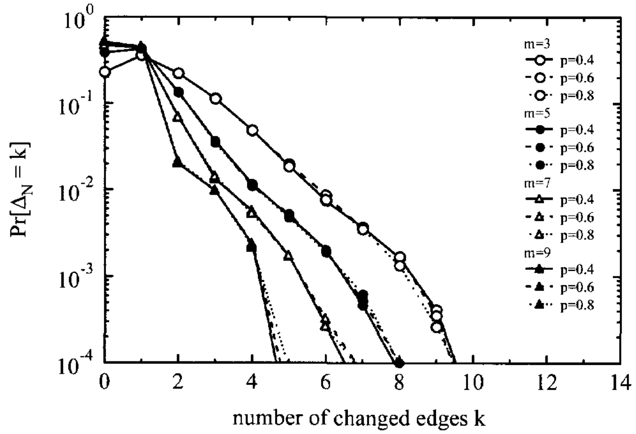


Figure 4.12: SMT: Pdf Pr $[\Delta_N = k]$ for $N = 10$ $(\alpha = 0.2)$.

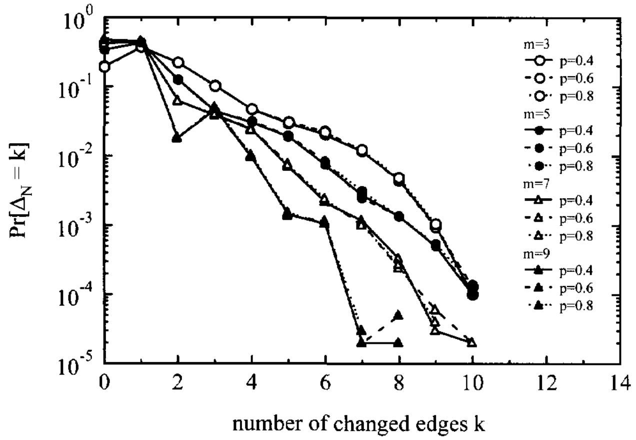


Figure 4.13: SMT: Pdf Pr $[\Delta_N = k]$ for $N = 10$ $(\alpha = 0.5)$.

where 1_x is the indicator function[5], we have simulated the pdf $\Pr[\Delta_{10} = k]$ as shown in Figures 4.12 to 4.17 for $a = 0.2, 0.5, 1, 2, 5, \infty$. The class RGU corresponds to $\alpha = 1$ and the last case ($\alpha = \infty$) corresponds to $w = 1$ on all links in the network.

The first observation from these figures is that the pdf $\Pr[\Delta_{10} = k]$ appears to be independent of the link probability p for $\alpha \leq 1$. Second, the larger $y = \frac{m}{N}$, the more correlation there is in the Steiner Tree which is reflected by oscillatory behavior of the probability density function. Third, these oscillations are more pronounced for the increasing power exponents α.

If the power exponent α is small (but $\alpha > 0$), $var[w] = \frac{\alpha}{(\alpha+2)(\alpha+1)^2}$ is relatively large (with a maximum for $\alpha = \frac{\sqrt{5}-1}{2}$ which is the "golden number") while $E[w] = \frac{\alpha}{\alpha+1}$ is small. This variation implies the existence of smaller link weights that will play a dominant role in the Steiner Minimum Tree. Since the Steiner Minimum Tree is a minimum link weight tree, the links with smaller weights will more likely be included in both the Steiner Minimum Tree with m and $m + 1$ multicast users. This will lead to a reasonable stable situation which is similar to the Shortest Path Tree dynamics. The larger part of the tree will not change if a multicast user leaves or joins. The number of changed branches Δ_N in the Steiner Minimum Tree is very unlikely to be smaller than in the corresponding Shortest Path Tree because by choosing a longer hop path, it may be possible to achieve a lower total weight of the tree. As a second implication of small α, the link weights have a thinning effect on the topology and overshadow the influence of the link density p: even if there is a link, it is the link weight that determines the importance of that link especially in shortest link weight problems. This explains the negligible effect of p as observed in Figure 4.12, 4.13 and 4.14. When α is large, $var[w] \to 0$ and $E[w] \to 1$. Let us consider the limit case of $\alpha \to \infty$. All links are equally important and, hence, the effect of the topology quantified by the link density p is important. If $p \to 1$, then $G_p(N) \to K_N$ and the behavior of Δ_N in the complete graph K_N with $w = 1$ is readily analyzed. Any Steiner Minimum Tree $s(m + 1)$ in K_N connecting $m + 1$ multicast users consists of precisely m links while the total link weight of that tree also equals m. Moreover, there exists a large number of different Steiner Trees. In particular, the number of different minimum spanning trees or $s(N)$ trees in K_N is precisely $(N - 1)$. The number of changed branches Δ_N consists of the total number of branches in $s(m + 1)$ and $s(m)$ minus the 2 times the number L_c of links in common. Hence, $\Delta_N = 2m - 3 - 2L_c$ or Δ_N is always odd, which explains the oscillatory behavior between odd and even values for Δ_N in Figures 4.16 and 4.17, especially for p high. The stability of these Steiner Minimum Trees is as worse as can be: the Steiner Tree $s(m + 1)$ in K_N may consist of entirely different branches from those of the Steiner $s(m)$ as exhibited by the wild oscillations in Figure 4.17.

In conclusion, the simulations have shown that the link weight distribution has profound influence on the stability of the Steiner Minimum Tree. The more links are

---

[5] The indicator function 1_x equals 1 if the condition x is true, otherwise it is zero.

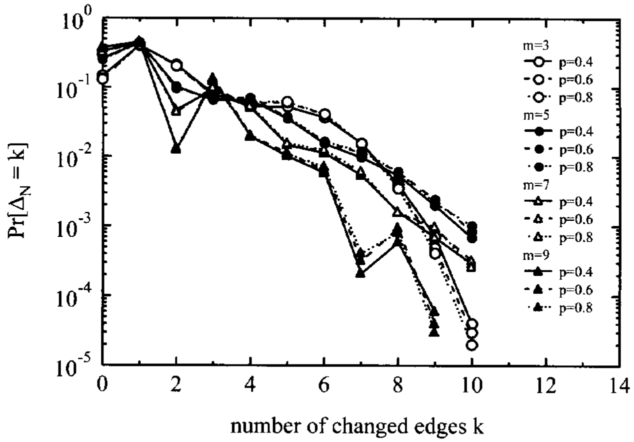


Figure 4.14: SMT: Pdf $\Pr[\Delta_N = k]$for $N = 10$ ($\alpha = 1$).

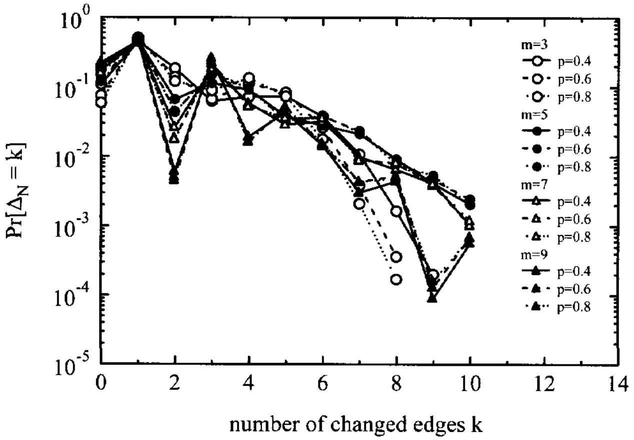


Figure 4.15: SMT: Pdf $\Pr[\Delta_N = k]$ for $N = 10$ ($\alpha = 2$).

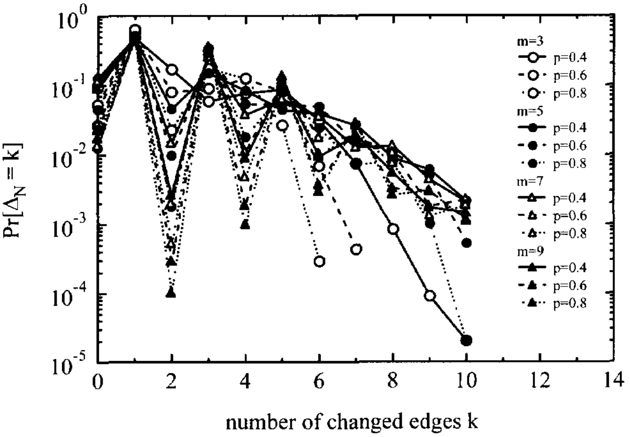


Figure 4.16: SMT: Pdf $\Pr[\Delta_N = k]$ for $N = 10$ $(\alpha = 5)$.

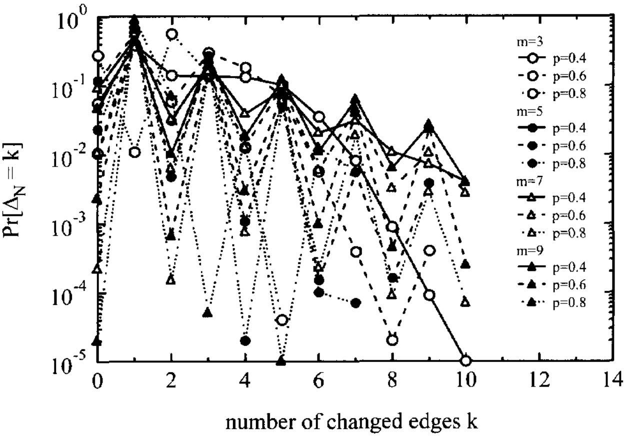


Figure 4.17: SMT: Pdf $\Pr[\Delta_N = k]$ for $N = 10$ $(\alpha = \infty)$.

equal (equivalent to α large), the higher the instability. If more links have different link weights, the more stable the Steiner Tree is. Whereas the underlying topology is decisive in the former, it plays hardly a role in the latter situation. Thus, the more the link weight structure of a network is heterogeneous, the healthier it is for the stability of the Steiner Trees. Recall the opposite behavior for the Shortest Path Tree as illustrated in Figure 4.10.

Influence of the size N of the graph. If we compare the results for the pdf obtained for $N = 10$ and $N = 20$ in the class RGU as illustrated in Figure 4.18, we observe that the probability density function for $N = 10$ and $N = 20$ match each other well for $y = \frac{m}{N} > 0.7$.

The mean $E[\Delta_N]$ and the variance $var[\Delta_N]$ were also computed and plotted as a function of the ratio $y = \frac{m}{N}$ in Figure 4.19. We observe for the class RGU ($\alpha = 1$) that the mean value seems independent of the number of nodes in the network, although the variances differ.

4.2.3 Comparison of Steiner and Shortest Path Tree

In order to compare the stability of the Shortest Path Tree (SPT) and the Steiner Minimum Tree (SMT) in the class RGU, we have plotted in Figures 4.20 and 4.21, the probability density functions of changed number of edges Δ_N for $N = 10$ and $N = 20$ nodes, and in Figures 4.22 and 4.23 the mean value and the variance of these pdfs. From these figures, the following observations can be made: (a) The maximum number of changed edges Δ_N in SPT does not increase with the increase of N as fast as for the Steiner Tree (SMT). This phenomenon has been explained previously: the minimization of the weight of the total tree forces the Steiner Tree to include longer hop paths if the sum of their link weights is smaller. (b) The pdf of Δ_N for the Steiner Minimum Tree possesses a larger tail which agrees with the common intuition that Steiner Trees are less stable than Shortest Path Trees. (c) The larger tail for the Steiner Tree also causes that the mean $E[\Delta_N]$ of SMT is larger than that of the SPT and similarly for the variance. (d) The more remarkable observation is that the mean $E[\Delta_N]$ for $N = 10$ and $N = 20$ in both SMT and SPT, hardly changes with N for nearly all values of $y = \frac{m}{N}$. Most likely, for RGU or $\alpha = 1$, the dynamics of the Steiner Minimum Tree resembles those of the SPT as argued above. The equality of $E[\Delta_N]$ and $var[\Delta_N]$ in SPT follows from the Poisson law (4.7).

4.3 Cost of Multicast Routing Trees

In the previous section we have demonstrated that Steiner Trees exhibit less stable dynamic behavior compared to Shortest Path Trees. This characteristic, in addition

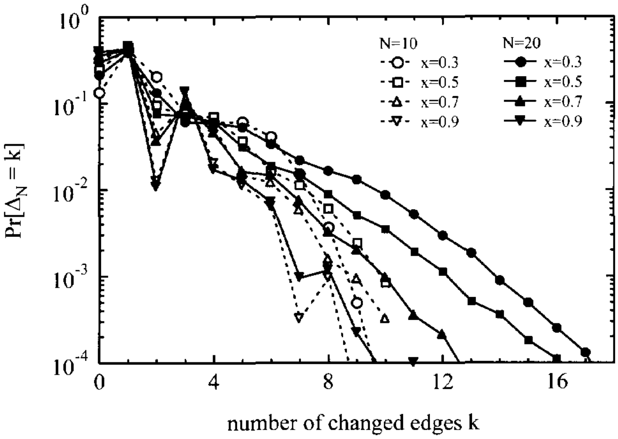


Figure 4.18: SMT: Pdf $\Pr[\Delta_N = k]$ for $N = 10$ and $N = 20$.

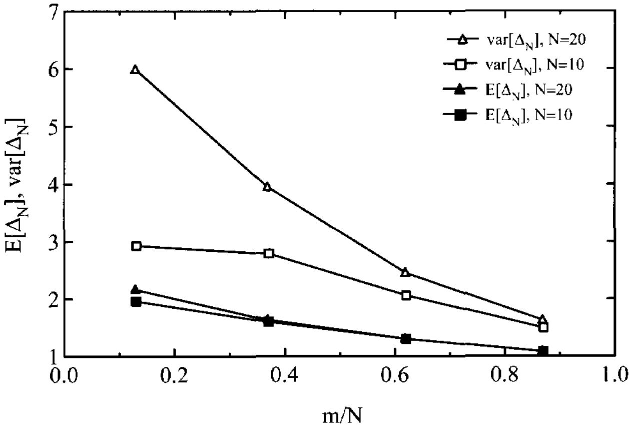


Figure 4.19: SMT: Mean and variance of Δ_N for $N = 10$ and $N = 20$.

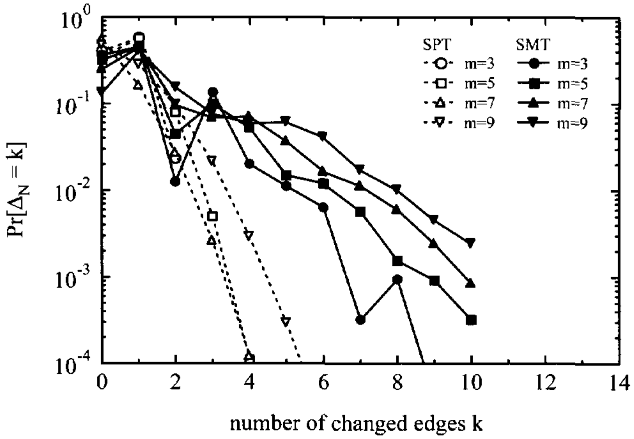


Figure 4.20: Comparison SPT and SMT ($N = 10$).

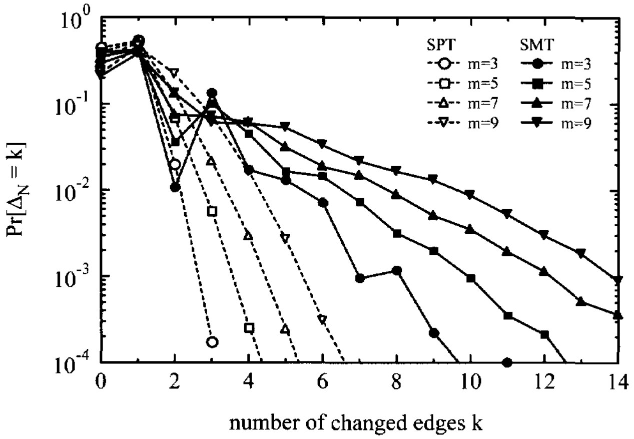


Figure 4.21: Comparison SPT and SMT ($N = 20$).

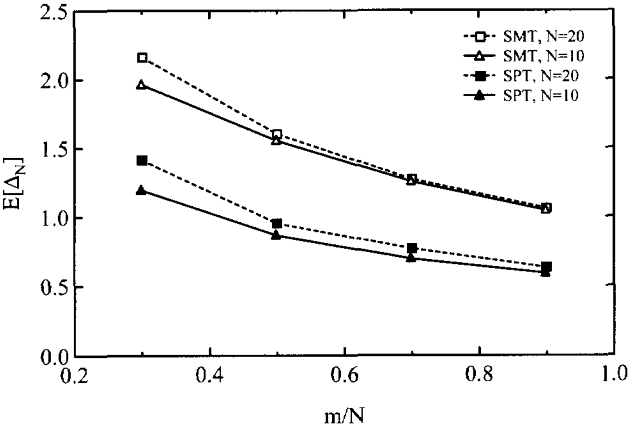


Figure 4.22: Mean of Δ_N in SMT and SPT ($N = 10, 20$).

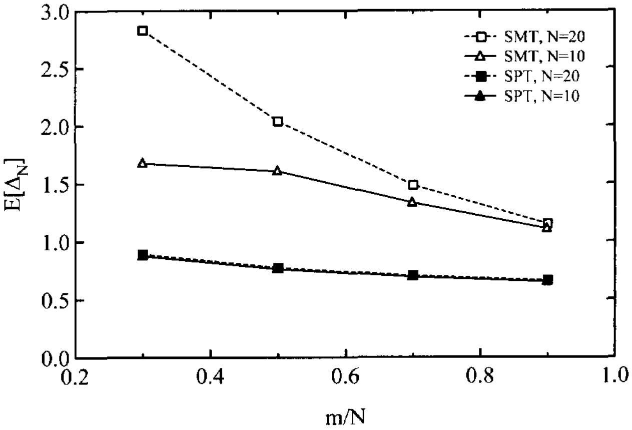


Figure 4.23: Variance of Δ_N in SMT and SPT ($N = 10, 20$).

to their computational complexity, prohibits the implementation of this algorithm in multicast routing protocols. Instead, most of the current multicast protocols forward packets based on the (Reverse) Shortest Path. The SPT algorithm does not necessarily result in a tree that economizes on network resources but it is easy to compute and it offers a minimum delay. The question that arises then is, how much more do the Shortest Path Trees cost, in terms of the total resources used, compared to the Steiner Trees.

Recently, Bollobas *et al.* [19] have shown that in the complete graph with N nodes and with exponentially distributed link weights with mean 1, the asymptotic weight of the Steiner Minimum Tree spanning $m+1$ uniformly chosen nodes and m small with respect to N,

$$E\left[W_{Steiner}\left(k\right)\right] = \left(1 + o\left(1\right)\right)\frac{m}{N}\log\frac{N}{m+1} \tag{4.8}$$

Triggered by this result, van der Hofstad *et al.* [99] have derived the average $E\left[W_N\left(m\right)\right]$ of the sum of the weights $W_N\left(m\right)$ in the SPT to m uniform multicast users in the random graph $G_p\left(N\right)$ with exponentially distributed link weights, expression (2.9). However, no analytic expressions are known for the distribution $\Pr\left[W_N\left(m\right) \leq x\right]$, nor for the corresponding probability generating function $\varphi_{W_N(m)}\left(z\right) = E\left[\exp\left(-zW_N(m)\right)\right]$. Here, we complement the analytical results derived in [99]. The significance of $W_N\left(m\right)$ for multicast is that $W_N\left(m\right)$ can represent the cost of used resources of the multicast tree, defined as the sum over all links in the multicast tree of the (monetary) costs of the resources used per link.

We proceed with presenting the simulation results and a conjecture on the probability density function of the weights. So far, we are not able to prove whether the probability density function of the normalized variable $W_N(m)$ tends to a Gaussian or to a Gumbel.

4.3.1 Cost: Simulation Results and a Conjecture

We have performed a set of simulations to complement the analytical results derived in [99] on complete graphs (with exponentially distributed link weights with mean 1). Therefore, in simulations presented in this section we confine ourselves to that class of graphs. For each number of nodes N, 10^5 topologies were generated randomly. For each of these topologies, $m \in \{1, ..., N-1\}$ nodes were uniformly chosen. The Shortest Path Tree (SPT) and the Steiner Minimum Tree (SMT) have been computed subsequently. The SPT is computed by using the Dijkstra algorithm, with $N \leq 100$. Depending on m, the Steiner Minimum Tree [45] is generated using different algorithms, as explained in Section 2.2. Again, like in our study on the stability, in spite of the implemented reductions, the simulation process is still extremely time consuming for large N. Therefore, we restrict the simulations of SMT to graphs with $N \leq 20$. In each graph and for each m, the sum of the weights as well as the number of links in both the SPT and the SMT have been stored in 4 histograms. From these histograms, the probability density function of the sum of the link weights $f_{W_N(m)}(x) = \frac{d}{dx}\Pr[W_N(m) \le x]$ in the SPT and the SMT, as well as the probability density function of the number of links $\Pr[H_N(m) = k]$ have been deduced. Since $\Pr[H_N(m) = k]$ is analytically known [99], these simulations are not shown, but served as a verification for the simulations.

Figure 4.24 gives the probability density function of the sum of the link weights $f_{W_N(m)}(x)$ in the SPT.

The average value $E[W_N(m)]$ of the sum of the link weights in the SPT and the SMT is plotted as a function of the number of multicast receivers m, for the number of nodes $N = 20$ in Figure 4.25. Apart from the match between simulations and theory for the SPT, this figure reveals that $E[W_N(m)]$ for the SMT seems similar (apart from some scaling factor) to that of the corresponding SPT. In Figure 4.26, simulation results of the variance of $W_N(m)$ in the SPT and the SMT are shown. So far, $var[W_N(m)]$ has only been derived analytically for $m = N - 1$.

Although N is small (which allows us to show the entire m-range), Figure 4.27 indicates that the scaled random variable $X_N(m) = \frac{W_N(m) - E[W_N(m)]}{\sqrt{var[W_N(m)]}}$ is close to a Gumbel type $e^{-e^{-x}}$, which may suggests, for all m, that

$$\lim_{N\to\infty} \Pr[X_N(m) \le x] = e^{-e^{-\frac{\pi}{\sqrt{6}}x-\gamma}} \tag{4.9}$$

where $\gamma = 0.5772...$ is Euler's constant. For the particular case of $m = 1$, we are able to prove this result (see below). However, simulations for larger N ($N > 1000$) seem to indicate that $X_N(m)$ tends to a normalized Gaussian for $m > 1$. As a matter of fact, for $m = N - 1$, convergence of $X_N(N-1)$ to a normalized Gaussian can be proved[6]. Figure 4.28 illustrates this behavior. Hence, $X_N(m)$ converges only slowly towards its asymptotic limit, implying that simulations are not the best device to obtain the asymptotic distribution.

Conjecture 1 *For all $m < m_c$,*

$$\lim_{N\to\infty} \Pr\left[\left(\frac{W_N(m) - E[W_N(m)]}{\sqrt{var[W_N(m)]}}\right) \le x\right] = e^{-e^{-\frac{\pi}{\sqrt{6}}x-\gamma}} \tag{4.10}$$

For the particular case of $m = 1$, we are able to prove this result:

Proof. Proof of Conjecture (4.9) for $m = 1$ ■

---

[6] R. van der Hofstad, G. Hooghiemstra and P. Van Mieghem, "The weight of the shortest path tree", unpublished.

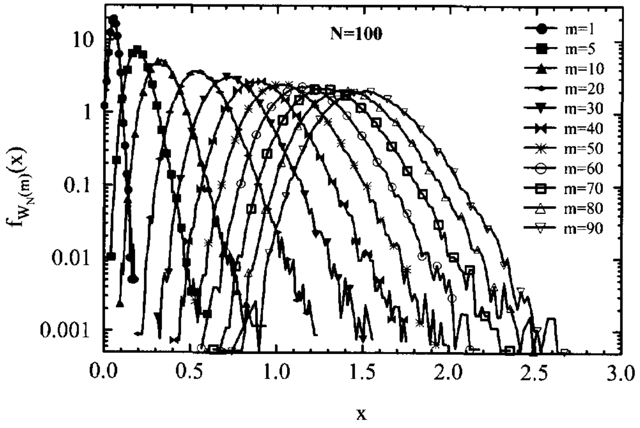


Figure 4.24: The pdf of sum of the weights in the SPT for $N = 100$.

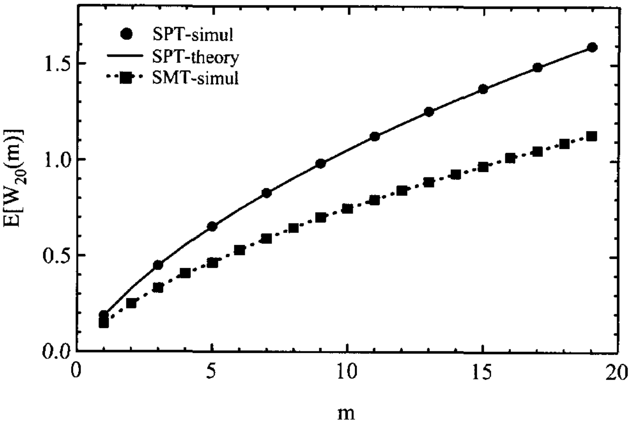


Figure 4.25: The average value of the sum of the link weights for SPT and SMT ($N = 20$).

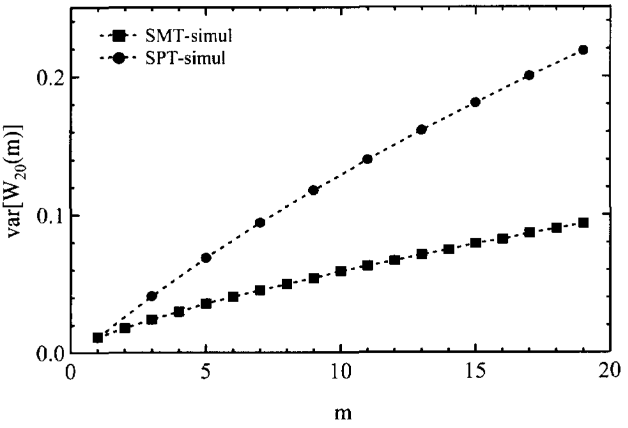


Figure 4.26: The variance of the sum of the link weights for SPT and SMT ($N = 20$).

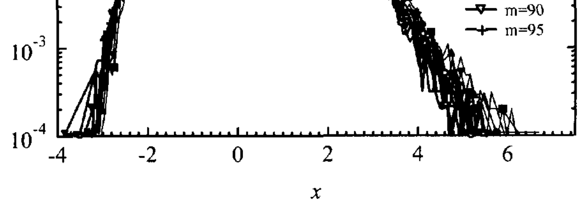


Figure 4.27: The pdf of the scaled random variable $X_N(m) = \frac{W_N(m)-E[W_N(m)]}{\sqrt{var[W_N(m)]}}$.

In [99], the probability generating function of the weight $W_N = W_N(1)$ of the shortest path is derived

$$\varphi_{W_N}(z) = E\left[e^{-zW_N}\right] = \frac{1}{N-1}\sum_{k=1}^{N-1}\prod_{n=1}^{k}\frac{n(N-n)}{z+n(N-n)}$$

The variance, computed from $\varphi''_{W_N}(0) - \left(\varphi'_{W_N}(0)\right)^2$, is

$$var\left[W_N\right] = \frac{3}{N(N-1)}\sum_{n=1}^{N-1}\frac{1}{n^2} - \frac{\left(\sum_{n=1}^{N-1}\frac{1}{n}\right)^2}{(N-1)^2 N}$$

and for large N,

$$var\left[W_N\right] = \frac{\pi^2}{2N^2} + O\left(\frac{\log^2 N}{N^3}\right)$$

The limit for $N \to \infty$ will be computed, from which the distribution then follows by taking the inverse Laplace transform. With $y = \sqrt{\left(\frac{N}{2}\right)^2 + z}$, we have

$$\prod_{n=1}^{k}\frac{n(N-n)}{z+n(N-n)} = \frac{k!(N-1)!}{(N-k-1)!}\prod_{n=1}^{k}\frac{1}{\left(y+\frac{N}{2}-n\right)\left(y-\frac{N}{2}+n\right)}$$

The products can be written in terms of the Gamma function,

$$\varphi_{W_N}(z) = (N-2)!\frac{\Gamma\left(y-\frac{N}{2}+1\right)}{\Gamma\left(y+\frac{N}{2}\right)}\sum_{k=1}^{N-1}\frac{\Gamma(k+1)}{\Gamma(N-k)}\frac{\Gamma\left(y+\frac{N}{2}-k\right)}{\Gamma\left(y-\frac{N}{2}+k+1\right)}$$

Let the number of nodes be even $N = 2M$ such that $y = \sqrt{M^2+z} \sim M + \frac{z}{2M}$ (provided $|z| < 2M$). The sum, denoted by S, can be rewritten

$$S = \sum_{j=-(M-1)}^{M-1}\frac{\Gamma(y+j)}{\Gamma(M+j)}\frac{\Gamma(M-j+1)}{\Gamma(y-j+1)}$$

such that

$$\begin{aligned}\varphi_{W_{2M}}(z) &= (2M-1)!\frac{\Gamma(y-M+1)}{\Gamma(y+M)} \\ &\times \frac{1}{2M-1}\sum_{j=-(M-1)}^{M-1}\frac{\Gamma(y+j)}{\Gamma(M+j)}\frac{\Gamma(M-j+1)}{\Gamma(y-j+1)}\end{aligned}$$

For large M,

$$(2M-1)!\frac{\Gamma\left(y-M+1\right)}{\Gamma\left(y+M\right)} \sim (2M)^{-\frac{z}{2M}}\,\Gamma\left(\frac{z}{2M}+1\right)$$

which makes us consider $z \rightarrow 2Mz$ since then, using [7, 6.1.47],

$$\begin{aligned}\varphi_{W_{2M}}\left(2Mz\right) &\sim (2M)^{-z}\,\Gamma\left(z+1\right)\\ &\times \frac{1}{2M-1}\sum_{j=-(M-1)}^{M-1}\left(1+O\left(\frac{1}{M}\right)\right)\\ &\sim (2M)^{-z}\,\Gamma\left(z+1\right)\end{aligned}$$

Hence,

$$\lim_{N\to\infty} N^{z}\varphi_{W_N}\left(Nz\right)=\Gamma\left(z+1\right)$$

or equivalently,

$$\lim_{N\to\infty} E\left[e^{-(NW_N-\log N)z}\right]=\Gamma\left(z+1\right) \tag{4.11}$$

The inverse Laplace transform of $\Gamma\left(z+1\right)$ is a Gumbel distribution. Since $E\left[W_N\right] \sim \frac{\log N}{N}$ and $\sqrt{var\left[W_N\right]} \sim \frac{\pi}{\sqrt{2}N}$, we arrive at the asymptotic distribution (4.9) for the weight of the shortest path ($m = 1$).

4.4 Implications for the Network Providers

Since multicast is more complex and more difficult to manage than unicast, an ISP finds it cost effective to deploy and manage it for customers, only when doing so saves significant bandwidth. Some work on establishing multicast business model has been done previously. However, most of these studies focus on practical issues of multicast pricing and billing, whereas the fundamental ISP's dilemma remains neglected [42, 57, 90, 6, 96]. For example, Einsiedler *et al.* [42] propose a scheme to charge the users for the resources they use. The resources are represented by the weights of the links, that are determined by the maintenance costs, congestion on the link, or other factors. At each router in the tree where branching takes place (branching point), costs are determined by splitting the cost among subtrees. Each router stores the information on the number of branches and on link weights, and distributes this information along the branches to other branching points, and finally toward the receivers.

We believe that charging users extra for using multicast is not justified. The technology used should be transparent to users. Moreover, users could be offered lower fees when using multicast in order to stimulate wide-spread usage. On the other hand, the ISPs could receive more revenues from content providers for offering multicast, since multicast enables content providers to reach more customers.

Instead, we focus solely on the extra costs and savings that employing multicast induces. The savings of multicast must outgrow the increased cost: the reduction in the network cost must be higher than the increase in management and deployment cost. If with C^m and C^u we denote the cost of multicast and unicast respectively, then

$$C^m = C_N^m + C_d^m + C_m^m$$

$$C^u = C_N^u + C_d^u + C_m^u$$

where C_N^m, C_d^m, C_m^m and C_N^u, C_d^u, C_m^u, are the network, deployment and management costs for multicast and unicast respectively, and the extra costs multicast induces compared to unicast are defined as

$$\Delta C = C^m - C^u = C_N^m - C_N^u + C_d^m - C_d^u + C_m^m - C_m^u \tag{4.12}$$

It is only if $\Delta C < 0$ that the network operators would consider deploying multicast.

Given that virtually all routers nowadays support multicast, and no infrastructure changes are needed, we can assume that

$$C_d^m - C_d^u \approx 0$$

Due to the complexity of computing the management costs exactly, some assumptions and estimates need to be made. We know[7] that an average ISP needs to employ approximately one to three engineers more to manage multicast. Hence,

$$C_m^m - C_m^u \approx C_E$$

where C_E represents a monthly cost of extra manpower needed to manage multicast.

Finally, the network costs can be expressed in the form:

$$C_N^m - C_N^u = \left(E\left[H_N\left(m\right)\right] - mE\left[H_N\right]\right) BC_n N_t$$

where m is the number of multicast receivers, $E\left[H_N\left(m\right)\right] = g_N(m)$ is the multicast efficiency given with (4.2), $E\left[H_N\right] = E\left[H_N\left(1\right)\right]$ is the average number of links to a uniform location in the graph given with (4.4), B is a bandwidth of an application, C_n is a monthly cost of transport of a 1 kB/s flow between two points of presence, and N_t a number of simultaneous multicast sessions in the network.

So, expression (4.12) becomes:

$$\Delta C = \left(E\left[H_N\left(m\right)\right] - mE\left[H_N\right]\right) BC_n N_t \ + C_E \tag{4.13}$$

Most of the ISPs today have to invest 10 kEuro for the monthly costs of an extra engineer, and they have to pay $C_n = 0.05$ Euro for the monthly cost of transport of kB/s of data over the network[8]. Furthermore, data published in [1] indicate that, currently,

---

[7] Private communication with several IPSs.

[8] Private communication with ISPs.

the number of simultaneous sessions is around $N_t \approx 100$ (the maximum number that can be supported in routers reaches approximately 20000). It is reasonable to assume that, on average, the costs for the extra manpower fall in the range C_E =[5000, ..., 20000] Euro, normalized monthly network costs in range C_n =[0.01, ..., 0.5] Euro, and that the number of simultaneous sessions varies from 1 to 20000. In addition, since ΔC depends on the bandwidth B required for a particular application, we can examine several possible multicast scenarios, for three different values of bandwidth (three different types of applications):

(*a*) 20 kB/s
(*b*) 128 kB/s for low quality video or high quality audio
(*b*) 1 MB/s for high quality video.

By substituting the above chosen values for parameters C_E, C_n, N_t and B in (4.13) and letting $\Delta C = 0$, we can obtain the break-even number of receivers m for which the additional costs of implementing multicast over unicast becomes 0, as a function of parameters C_E, C_n, N_t and B.

Figures 4.29, 4.30 and 4.31 display the break-even number of multicast receivers m , as a function of C_E, C_n and N_t, respectively. In each figure, three lines have been plotted, each corresponding to a different value of bandwidth B. The vertical line refers to most likely values of parameters C_E, C_n and N_t (10000, 0.05, and 100 respectively).

Provided our estimates are correct, and given that, on average, 10% of subscribers participate in multicast, we obtain that multicast pays off for network providers

(a) approximately 300 subscribers in case of 20 kB/s flows
(b) approximately 90 subscribers for 128 kB/s
(c) approximately 20 subscribers for 1 MB/s flows

4.5 Conclusions

This chapter presented our study of the behavior and the properties of multicast routing trees, both analytically and via simulations. First, we have reviewed the analytical results for the savings of multicast over unicast, in terms of the number of traversed links (hops), derived earlier in [101], valid for any type of network topology.

Further, the stability of the routing trees, defined as the number of links that changes in the tree when one user joins or leaves the group, has been analyzed. The stability of both the Shortest Path Tree (SPT) and the Steiner Tree (SMT) has been quantified for the class of random graphs $G_p(N)$. The Poisson law (4.7) for the number of changed links Δ_N in SPT, has been proven mathematically for the class $G_p(N)$, while simulation results point towards a larger applicability of the Poisson law than the asymptotic regime. In addition, we have argued that similar laws as the Poisson law can be obtained for a general topology (including that of the Internet), provided both the hopcount distribution $\Pr[H_N = k]$ and the multicast efficiency $g_N(m)$ are known. Hence, the

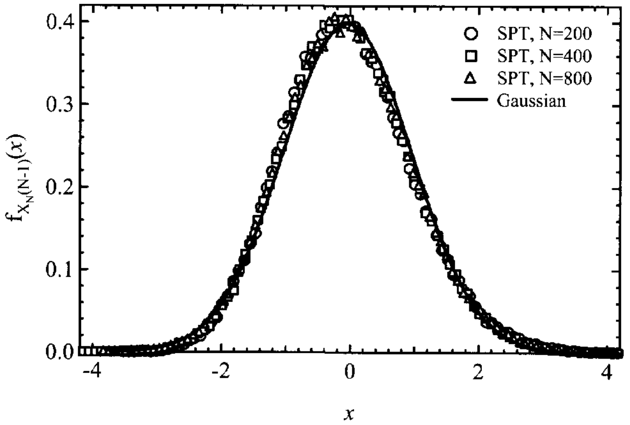


Figure 4.28: The convergence of $X_N\,(N-1)$ to a normalized Gaussian.

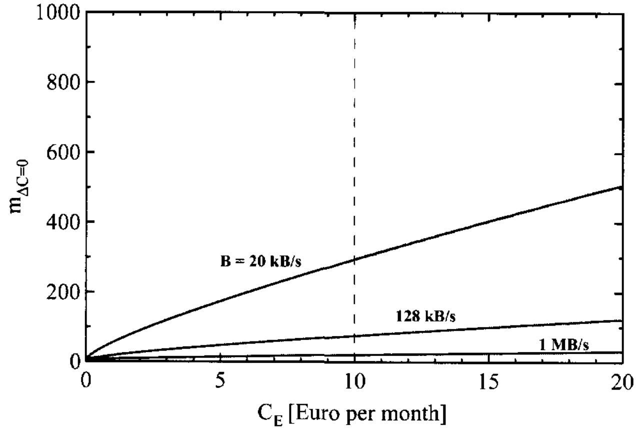


Figure 4.29: The break-even number of multicast receivers m as a function of menpower cost C_E, with bandwidth B as a parameter ($B = 20$ kB/s, 128 kB/s and 1 MB/s).

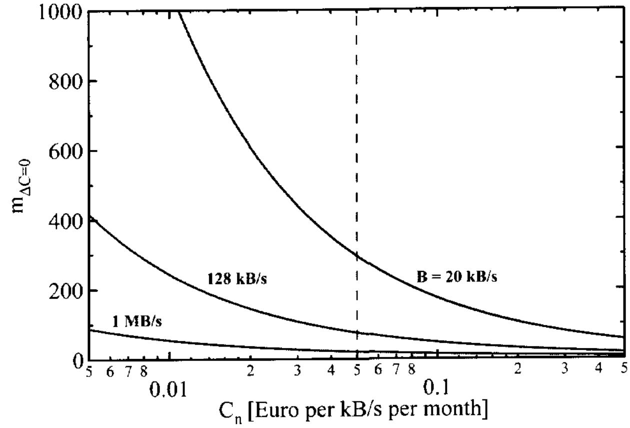


Figure 4.30: The break-even number of multicast receivers m as a function of normalized network costs C_n, with bandwidth B as a parameter ($B = 20$ kB/s, 128 kB/s and 1 MB/s).

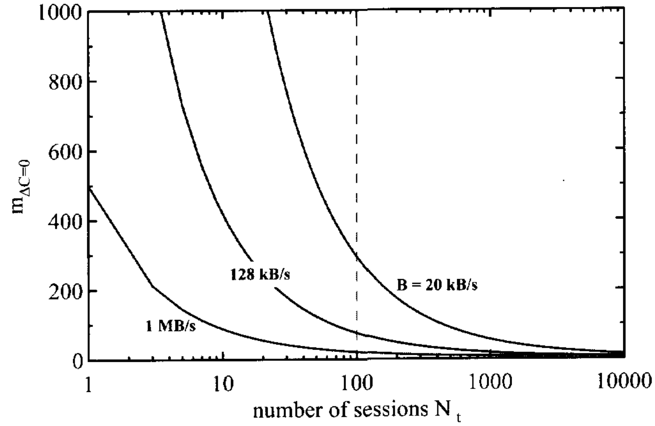


Figure 4.31: The break-even number of multicast receivers m as a function of the number of simultaneous multicast sessions N_t, with bandwidth B as a parameter ($B = 20$ kB/s, 128 kB/s and 1 MB/s).

stability (in our setting) of the Shortest Path Tree problem may be regarded in principle as approximately solved.

The behavior of the Steiner Minimum Trees is not entirely understood and requires further analysis. Especially, for large N, it would be interesting to find the scaling laws of the Steiner Minimum Tree as well as the tail behavior. Apart from large network sizes N, the simulations show that the link weight distribution determines the stability of the Steiner Tree problem. If the majority of the links have different weights, the stability of the Steiner Minimum Tree resembles that of the Shortest Path Tree. The other extreme, where most link weights are equal, leads to large instabilities reflected by wild oscillations in the corresponding pdf $\Pr[\Delta_N = k]$. The stability of the Steiner Minimum Tree is in most situations worse than that of the corresponding Shortest Path Tree. Mainly because the departure or arrival of a multicast member may cause other branches to be included in the Steiner Minimum Tree (to achieve an overall minimum in the sum of the weights) than just the branches of the shortest path towards the subtree rooted at B (as defined in Figure 4.1).

We conclude that the intuitive assumption on the (in)stability of Steiner Minimum Trees is correct. Even though Steiner Trees optimize the use of resources, they cannot be used in multicast protocols. If we define the cost as the sum of used resources, then a question that arises is, how much more costly are the Shortest Path Trees compared to Steiner Trees. With link weights representing the available resources, the sum of used resources can be expressed as the sum of the link weights in the tree. The sum of the link weights in the SPT is on average not more than 37% worse than that in SMT. From the extensive simulations we were lead to conjecture that the probability density function of the scaled sum of the link weights for small values of N tends to a Gumbel, however, with the increase of N, it converges slowly toward a Gaussian.

Finally, this chapter discussed some possible business scenarios for network operators. Among the additional costs of multicast over cost of unicast we distinguish the deployment, management and network costs. By computing the network costs, and estimating the deployment and management ones, we suggested that at moderate bandwidths (e.g. 128 kB/s), multicast becomes beneficial for network operators for approximately hundred receivers. For higher bandwidths (e.g. 1 MB/s) the break-even number of receivers is achieved with approximately ten receivers.

Chapter 5

Measurement-Based Analysis of Multicast Trees

The framework proposed and explicated in Chapter 4 allows us to exactly compute certain properties of multicast trees. This framework is valid for any number of multicast receivers m, any topology and any number of nodes N in the topology. Thus, it should also be valid for the topology of Internet. Nevertheless, in order to convince the network operators that the business model based on that framework is utilizable, we need to investigate how well it matches the real Internet data.

Understanding and modeling the topology of the Internet has attracted considerable attention in the last decade. Since the topology can influence the performance of network protocols deployed, a reliable topology model is needed in order to examine and verify their quality. Due to the volatile nature of the Internet, its exact reproduction is an impossible task. Still, through the measurements of (a part of) the Internet, we can capture some fundamental topological properties, such as the node degree distribution. Generating topologies that possess these properties would create topology models that closely resemble the Internet.

Both passive and active measurements are used for obtaining more insight into the topology of the Internet. The most popular tool for acquiring the map of the Internet thus far has been the *traceroute* utility [95]. However, as we will demonstrate further in this chapter, there are many ambiguities and imperfections related to *traceroutes*. Consequently, care is needed when extrapolating the results obtained via *traceroute* measurements to the Internet as a whole.

One of the most striking observations in the Internet maps is the long-tailed node degree distribution. It has first been observed by Faloutsos *et al.* [46] in 1999, when they published their empirically derived results on properties of several Internet topology parameters, among which the node degree distribution. They studied three different Internet AS level instances, as well as a router-level map. These findings have provoked astonishment among network researchers and stimulated the desire to explain this behavior. However, Chen *et al.* [30] have criticized the results published in [46] implying that they were obtained on incomplete AS graphs. They show that when data is obtained from more complete AS graphs, the degree distribution follows heavily skewed distributions (the values of degree vary over 3 to 4 orders of magnitude), but where only tails decay like a power-law. In this chapter we present a thorough analysis of results on node degrees obtained by the research community so far. We will show that, based on *traceroute* data, the node degree distribution in the Internet map does not always follow a power-law.

Oddly, whereas the structure of the Internet topology has been the focus of considerable study, modeling multicast trees has not received the attention it deserves. Only few results have been available on the node degree distribution of multicast routing trees, which provided contradictory conclusions. Similar to the node degree distribution in the Internet map, we will show that the node degree distribution in the multicast tree does not always follow a power-law.

In addition to the node degree distribution, we have investigated the number of links in multicast trees. In particular, we investigate how well Internet measurement data fit the model proposed in the previous chapter.

The remainder of this chapter is organized as follows: In the following section the collection of measurement data is explained. In Section 5.1.3 we discuss the ambiguities and inaccuracies in *traceroutes* and the dangers of drawing conclusions based on these data. In Section 5.2.4, we focus on a problem specific for multicast analysis of unicast *traceroute* measurements - the occurrence of cycles. In Section 5.3 we concentrate on node degree distributions of Internet maps. In Section 5.4 multicast trees structure has been discussed. Section 5.4.1 covers the related work on the node degree distribution of multicast trees. The *traceroute*-based multicast trees and the quality of the URT as a model for multicast trees on Internet is investigated in Section 5.4.2. There we present our results on the node degree distribution. In Section 5.4.3 the number of links in Internet multicast trees is evaluated. Conclusions are summarized in Section 5.5.

5.1 Measurement data sets

For our analysis on multicast trees and maps of the Internet, *traceroute* data was needed. *Traceroute* infers an IP path between a source and a destination by sending out probe packets with progressively increasing TTLs and then analyzing the ICMP error responses sent by routers along the path receiving a packet with a zero TTL. In this section we describe three measurement architectures, RIPE NCC, Caida and Planetlab, that we used to obtain *traceroute* data.

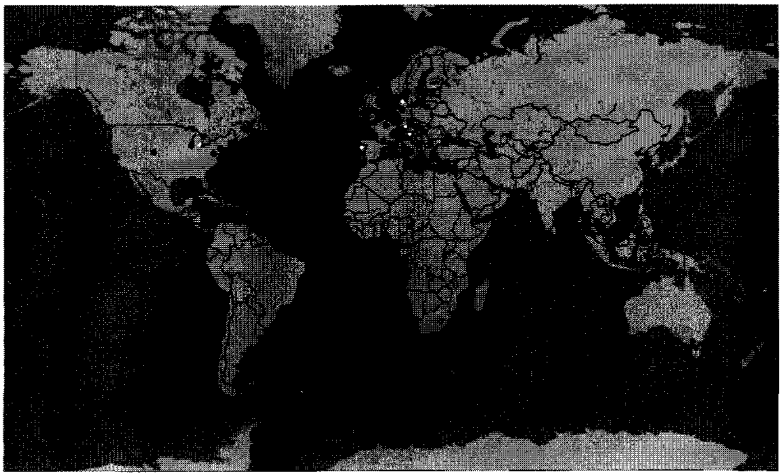


Figure 5.1: Distribution of RIPE testboxes over the world.

5.1.1 RIPE-NCC

RIPE NCC (the Network Coordination Centre of the Réseaux IP Européen) [5] performs *traceroute* measurements between measurement boxes scattered over Europe (and few in the US, Middle East, Japan and Australia/Oceania)(see Figure 5.1). At the time of writing, the number of boxes has been 92 with on average 2 boxes being added per month. We have collected the data in three periods: 1998-2001 (dataset1), May 1st - June 1st 2003 (dataset2) and January 1st - February 1st 2004 (dataset3).

The measurements are executed in the following way: approximately 10 times an hour, a *traceroute* is sent between each pair of measurement boxes, and the data has been collected once a day in a central point in RIPE (see Figure 5.2; for a further detailed description of the measurements and measurement configuration we refer to [5]). Subsequently, for each source-destination pairs, the *traceroute* data has been inserted into two tables in a database in the local computer in TUDelft. An example of these two tables is given in Figure 5.3. For the particular source-destination pair, the first table consists of two columns, *routeID* and *count*. To each unique path (a combination of IP-addresses visited by a packet traveling from the source to the destination test box) a unique *routeID* has been assigned. The number of times a particular route has been returned by *traceroute* has been stored in the *count* column. The second table specifies for each *routeID* the entire list of returned IP-addresses.

Since not all the boxes are active all the time, the number of boxes from which we obtained the data is smaller than the total number of boxes. In the period 1998-2001,

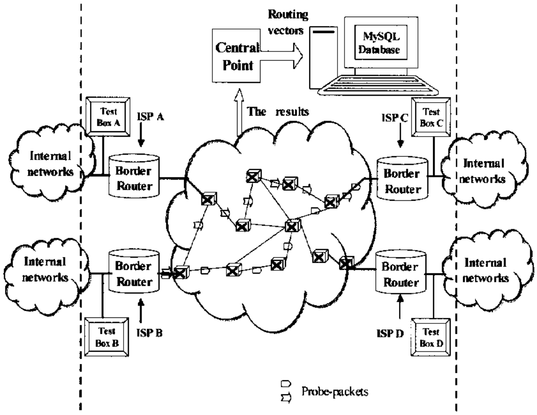
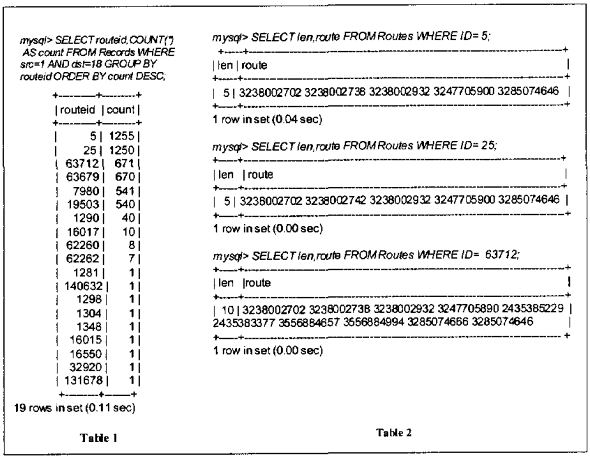


Figure 5.2: RIPE measurement configuration.

the total number of boxes has been 50. However, due to errors that will be discussed in the following section, 19 test-boxes have been excluded from further analysis. In the *traceroute* measurements among the remaining 31 active boxes the most dominant path, i.e. the path occurring most frequently, has been determined. In this way, a total of 465 most dominant paths has been gathered. However, we ascertained further that 17% of the those *traceroutes* suffer from errors, again to be addressed in the following section, leaving 386 non-erroneous dominant paths (dataset 1). In the period May 1st-June 1st 2003, from each of 61 active boxes, *traceroutes* to all the other test boxes have been obtained, resulting in a total of 1329019. Among all the distinguished paths in the database, only the most dominant one has been considered. After discarding the erroneous data, 973 non-erroneous most-dominant paths were distinguished (dataset 2). Finally, the same measurements have been executed in the period January 1st- February 1st 2004, and *traceroute* paths from 72 sources to a variable number of destinations (60–70) have been collected. The number of collected dominant non-erroneous *traceroutes* paths in this experiment was 4521 (dataset 3).

This collection of measurements we have used both for the creation of Internet router-level maps, as for the analysis of multicast trees. The data for the multicast analysis has been collected in the following way: each destination measurement box has been regarded as a multicast user. Multicast trees have then been constructed as the union of paths returned by *traceroutes*. Since PIM-SM (the most commonly used multicast routing protocol) relies on unicast routing tables for routing, trees constructed as union of *traceroutes* will resemble multicast routing trees.



```
mysql> SELECT routeid,COUNT(*)
 AS count FROM Records WHERE
src=1 AND dst=18 GROUP BY
routeid ORDER BY count DESC;

+--------+-------+
| routeid | count |
+--------+-------+
|      5 |  1255 |
|     25 |  1250 |
|  63712 |   671 |
|  63679 |   670 |
|   7980 |   541 |
|  19503 |   540 |
|   1290 |    40 |
|  16017 |    10 |
|  62260 |     8 |
|  62262 |     7 |
|   1281 |     1 |
| 140632 |     1 |
|   1298 |     1 |
|   1304 |     1 |
|   1348 |     1 |
|  16015 |     1 |
|  16550 |     1 |
|  32920 |     1 |
| 131678 |     1 |
+--------+-------+
19 rows in set (0.11 sec)
```

Table 1

```
mysql> SELECT len,route FROM Routes WHERE ID= 5;
+-----+-----------------------------------------------------------+
| len | route                                                     |
+-----+-----------------------------------------------------------+
|   5 | 3238002702 3238002738 3238002932 3247705900 3285074646    |
+-----+-----------------------------------------------------------+
1 row in set (0.04 sec)

mysql> SELECT len,route FROM Routes WHERE ID= 25;
+-----+-----------------------------------------------------------+
| len | route                                                     |
+-----+-----------------------------------------------------------+
|   5 | 3238002702 3238002742 3238002932 3247705900 3285074646    |
+-----+-----------------------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT len,route FROM Routes WHERE ID=  63712;
+-----+-----------------------------------------------------------+
| len | route                                                     |
+-----+-----------------------------------------------------------+
|  10 | 3238002702 3238002738 3238002932 3247705890 2435385229    |
2435383377 3556884657 3556884994 3285074666 3285074646          |
+-----+-----------------------------------------------------------+
1 row in set (0.00 sec)
```

Table 2

Figure 5.3: Tables in the MySQL database.

5.1.2 CAIDA

The Cooperative Association for Internet Data Analysis (CAIDA) provides tools and analyses promoting the engineering and maintenance of a robust, scalable global Internet infrastructure. CAIDA's Skitter tool [2] deploys a method similar to *traceroute* to determine the IP path to a destination. Destinations are chosen from BGP tables and a database of Web servers. Skitter sends ICMP echo request packets, increments the TTL when sending them and registers the IP address of the replying routers. If a router does not respond to three subsequent ICMP request packets, the TTL is increased. When the desired destination is reached, Skitter registers the round-trip-time (rtt) as well. However, in case TTL becomes 30, or "ICMP unreachable" reply has been received, or a routing loop is encountered, Skitter stops probing the destination. Resolving the aliasing problem is attempted as well, by deploying the iffinder tool. The iffinder tool relies on a similar router identification technique as the one described in [52].

The CAIDA Skitter project has deployed around 30 monitors worldwide. Each monitor performs *traceroutes* measurements to thousands of destinations every day. We have obtained the *traceroutes* from 6 skitter monitors over 3 days measurements (1,2,3 April 2003). Totally 684135 *traceroutes* have been collected in our database. The incomplete traces have been eliminated, leaving 276680 out of 684135 complete and stable *traceroutes* in the database.

Multicast routing trees have been obtained in the following way: first, we have randomly chosen $m = 50, 100, 500, 1000, 2000, 5000, 10000, 20000$ destinations (multicast users). For three monitor boxes (two of them situated in United States and one in

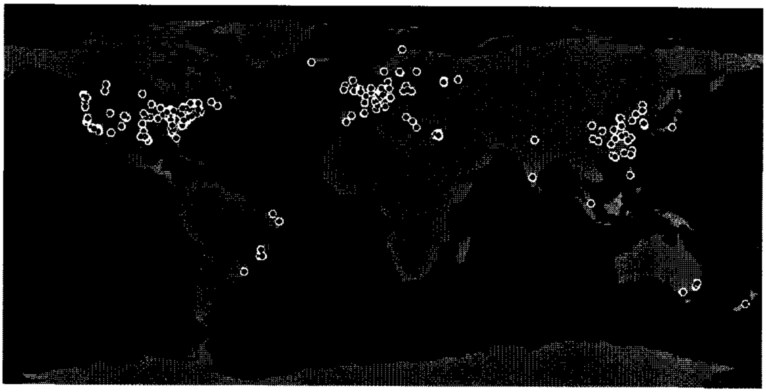


Figure 5.4: Current distribution of 537 PlanetLab nodes over 254 PlanetLab sites.

Japan) the collection of paths from these three sources to randomly chosen destinations has been obtained via *traceroute*. In this way, we obtained for each source a set of 8 trees. These trees resemble multicast routing trees, under the assumption that multicast group members are uniformly distributed. This assumption has been discussed in Chapter 4.

5.1.3 PlanetLab

PlanetLab [4] is an open, worldwide distributed testbed that enables performing experiments under real-world conditions, and on a large scale. At the time of writing, there were more than 250 institutions participating in PlanetLab projects, including TUDelft. Our experiments have been executed on November 10th 2004. At that moment, there were 445 PlanetLab nodes running on locations in USA, Asia and Europe (see Figure 5.4). Architecturally, the PlanetLab network is similar to RIPE: each node can serve as a source as well as a destination. From each of the PlanetLab sites, we can perform *traceroutes* to all the other PlanetLab sites. Since in some cases there are multiple nodes per PlanetLab site (situated at the same location), we selected one node per PlanetLab site, resulting in a total of 79 nodes (since many nodes are not active or not accessible at all time). Multicast trees have been generated in the same fashion as from RIPE measurement data.

Problems with *traceroutes*

The *traceroute* utility [95] has been the most popular tool for acquiring a map of the Internet so far. Once an extensive amount of *traceroutes* from multiple sources to multiple destinations is acquired, the Internet map can be created as a union of these *traceroute* paths. Nevertheless, there are many issues considering *traceroute* measurements that complicate this apparently straightforward procedure. These unsolved issues raise the question of the trustworthiness of conclusions based on *traceroutes*.

5.2 Ambiguities in *traceroutes*

5.2.1 Errors and inaccuracies

In spite of being the best current tool for inferring the end-to-end IP level path, *traceroutes* suffer from several types of errors and flaws [106]. In our dataset 1 (data from 1998-2001), we have analyzed the errors in paths returned by *traceroutes*, categorized them following the nomenclature proposed by Paxson in [78, 77], and compared to his results.

Unresponsive routers

A certain number of routers will not reply to *traceroute* probes. Totally, for dataset 1, consisting of 465 most dominant paths, 4 paths contained unresponsive routers, forming 0.86% of all the dominant paths' records.

Routing loops

A routing loop is diagnosed if a node with a certain IP address appears in one record more than once. We distinguish two types of loops, "persistent routing loops" and "temporary routing loops."

A loop is considered persistent, if *traceroute* records include loops that were not resolved by the end of the *traceroutes*, that is, after probing 30 hops. Out of 465 most dominant paths, persistent routing loop has been detected in only 1 of them, making 0.2% of all the records.

A routing loop is temporary if loops within routes are resolved, in other words, if the *traceroute* probe bypasses the loop and reaches the destination. An example of a temporary routing loop between two test-boxes is:

```
1    x.x.70.193
2    x.x.201.98
3    x.x.201.128
4    x.x.201.128
5    x.x.71.194
...
```

We notice that at hop 3 and hop 4, the same IP-address is detected. 35 out of the 465 most dominant *traceroutes* contained temporary routing loops, which makes 7.5% of all the records.

Compared to results obtained by Paxson [78, 77], the probability of temporary routing loops in data obtained from RIPE in 2001 increases largely. The difference is mainly caused by the definition of the unknown IP address (255.255.255.255) in RIPE According to this definition, a significant number of unknown IP addresses is mapped to the same 255.255.255.255, increasing the probability of temporary routing loops appearance. Approximately 60% of all temporary routing paths are caused by an unknown IP address (255.255.255.255).

Infrastructure failure

This failure reflects a problem inside the Internet infrastructure: the terminating router in the *traceroute* record was somewhere in the middle of the network, not at the destination node. Surprisingly, out of the 465 dominant *traceroutes*, 37 exhibited this failure, comprising 8% of all the records.

One possible cause is that some network sites have placed "firewalls" to filter incoming network traffic for security purposes. The firewall drops the packet without returning an ICMP Time Exceeded message. Thus, packets are lost and the source never receives a reply for a given hop.

Other inaccuracies

There are other types of flaws related to *traceroute* measurements as well. We have mentioned that some ISPs hide their routers from *traceroutes* by manipulating the ICMP replies. This can reduce the accuracy of topologies discovered. In addition, since two probes are sent to every router on the path, a considerable amount of overhead is generated. Finally, as reported in [94], when performing *traceroute* measurements using different tools (Skitter [2] and Rocketfuel [94]) in the same area of interest (with the time difference of two months) a noticeable number of different routers and links have been found (5-10 times as many addressess, links, routers, for a given ISP have been found with Rocketfuel, however some have been only found with Skitter).

5.2.2 Alias resolution

The *traceroute* utility returns the list of IP addresses of routers along the path from the source to the destination. One router can nevertheless have several interfaces, with several different IP addresses. The aliasing problem arises because an ICMP response packet of a router along a path has the address of the outgoing interface of the response as the source address, instead of the interface on which the probe packet triggering that response arrived. Since these two addresses can be different, when reading the *traceroute* responses, one would assume that they belong to two different routers. In order to obtain the accurate router-level maps, it is necessary to determine which IP addresses belong to the same router. This, however, is not a trivial problem. The early Internet mapping attempts have either ignored alias resolution [20], or have used a very simple alias probe heuristic [76]: after sending an alias probe packet to a non-existing port, the router responds with an ICMP port unreachable message. If the source address of this packet is different than the address to which it has been sent to, then these two addresses represent two interfaces belonging to the same router. The second is an alias resolution heuristic proposed by Govindan and Tangmunarunkit in Mercator [52] that relies on alias probing, but includes two refinements. The alias probe packets to all interfaces are repeatedly sent. A second refinement is necessary because large backbones do not have complete routing tables, and backbone(s) to which the sending host is eventually connected may not be able to forward an alias probe to its eventual destination. Therefore, Mercator [52] sends alias probes via source-route capable routers. Mercator cannot discover all interface addresses belonging to a router. Instead, it discovers only those interfaces through which paths from the Mercator host enter the router. The most enhanced alias resolving technique up to this moment, has been designed in the Rocketfuel project [94]. The pair-wise method implemented in Rocketfuel discovers three times more aliases than the preceding techniques. Rocketfuel includes Mercator's address-based heuristic, but combines it with other methods: it compares TTLs in response, tests ICMP rate limiting triggered by earlier probes, but most importantly, Rocketfuel compares the IP identifier field of the responses. Rocketfuel's IP identifier-based method significantly outperforms the Mercator heuristic: it finds almost three times as many aliases as an address-based method. Nevertheless, in spite of being the most effective alias resolving technique, Rocketfuel's Ally tool still encountered some substantial problems, such as unresponsive IP addresses (almost 6000 out of 56000 addresses did not react when queried for aliases).

5.2.3 Bias sampling

Lakhina *et al.* [69] have pointed to the bias in sampling when deducing topological properties of maps based on *traceroute* measurements. Currently, most of the *traceroute* measurements are performed from a limited number of publicly available sources, to a larger number of more flexibly chosen destinations. Consequently, nodes and links lying nearer to the sources will be visited much more frequently than those that are more remote, forming a possible cause for the manifestation of the long-tailed node degree distribution.

The same authors point to another significant problem: if a graph is generated by aggregating shortest paths from a limited number of sources and destinations, the node degree distribution in those graphs can vary significantly. They demonstrated via simulations that the random graphs of the class $G_p(N)$ [18], with Poisson node degree distribution, will appear to have a power-law characteristic. This implies that it is fallacious to characterize the router-level topology of Internet based on the node degree distribution of its (imperfect) subgraph. For all these reasons, in this chapter, we confine to analyzing the *traceroute* data, avoiding to impose the conclusion to the whole Internet router-level topology.

5.2.4 Analysis of cycles

The topology of a multicast tree is one of the significant elements for successful multicast management. Attaining this data is, however, a complex task, much more difficult than tracing unicast paths. The determination of the multicast tree requires accessing the data in the routers themselves. Considering the difficulty to acquire real multicast data and the rising need to analyze the multicast trees, we have assumed that paths constituting a multicast tree are actually the ones returned by *traceroute* measurements. The MBone era, consisting of tunnelling and DVMRP, is long behind us. Many network operators, the number raising continuously, implement native multicast today. Since PIM-SM (the most commonly used multicast routing protocol) relies on unicast routing tables for routing, the assumption we make seems justifiable.

However, when constructed as a union of *traceroutes*, the resulting topology can include cycles, and therefore is not a tree. Namely, when tracing paths from source X to m different destinations, in some cases the subsections of paths between the same two nodes consisted of different nodes. This problem is illustrated in Figure 5.5. In Figure 5.5(a), for three destinations that have been traced from a source 12, a path from the source 12 has traversed nodes 10 and 665. In two out of three cases *traceroute* has returned nodes 196 and 666 as intermediate nodes between 10 and 665, whereas in the *traceroute* record for the third destination, destination nodes 197 and 195 have been detected. In the real multicast session, the corresponding multicast tree would consist either of segment 10-196-666-665 or 10-197-195-665, but not of both of them. In our example, since the majority of paths (two out of three) have traversed the nodes 10-196-666-665, we assumed that this segment would appear in the multicast tree. Actually, the choice of either path in Figure 5.5(a), does not influence neither the degree distribution of the nodes, nor the total number of links in the tree. We call this type of cycles symmetrical. However, not all the loops detected in our data are symmetrical. The examples depicted in Figure 5.5(b) and 5.5(c) show asymmetrical loops. The cycle in Figure 5.5(c) is seemingly symmetrical, however the existence of children of the node 293 (an intermediate node in one segment) affects the total number of links and the degree distribution as well.

In the RIPE-NCC *traceroute* dataset 2, a total of 606 loops in 72 created trees has been detected, out of which 493 (81%) were symmetrical. The loops occur probably due to load balancing and router/link failures. The distribution of the number of loops per source node is shown in Figure 5.6.

In Figure 5.7 the hopcount distribution in the loops has been given suggesting that $\Pr[hop = k] \sim e^{-\alpha k}$, with $0.7 < \alpha < 1$, which implies that most loops are short.

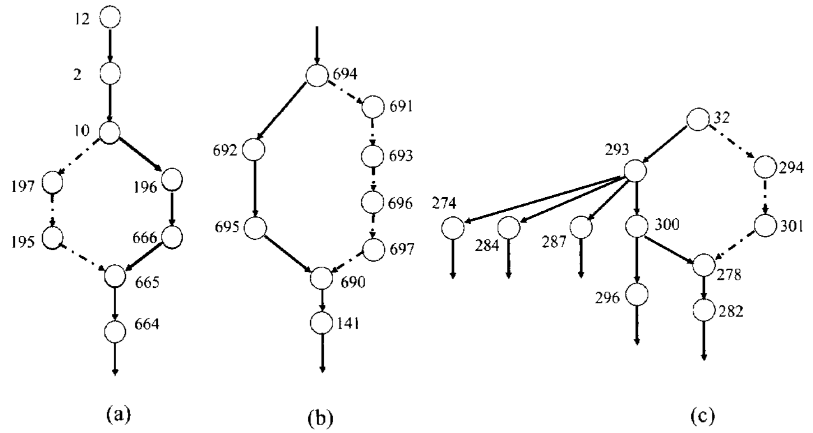


Figure 5.5: The illustration of cycles: (a) symmetrical, (b) asymmetrical type 1, (c) asymmetrical type 2.

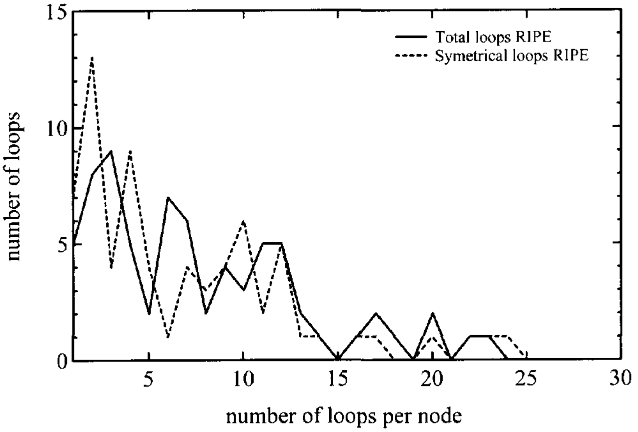


Figure 5.6: The number of loops per source node.

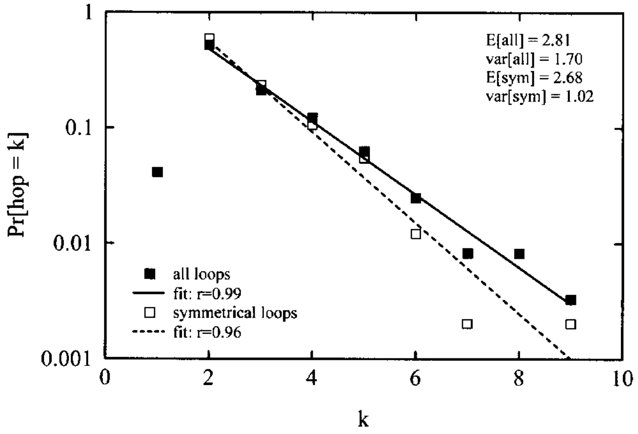


Figure 5.7: The distribution of the hopcount in the loops.

dataset	# traces	# srcs	# dests	#loops-aver.	# sym. loops-aver.
RIPE	4521	72	60-70	7.66	6.75

Table 5.1: The average number of loops and symmetrical loops per tree (source).

5.3 Node degree distributions in the maps of Internet

5.3.1 Overview of node degree distributions obtained in other research groups

One of the first attempts to map the Internet router-level topology was that of Pansiot and Grad in 1995. Pansiot and Grad have constructed the Internet map based upon the *traceroute* records from a single node to 5000 geographically dispersed destinations, as well as on *traceroutes* from a subset of 11 nodes chosen from the set of 5000 nodes to the rest of the destinations. Based on these records, they created a graph containing 3888 nodes and 4857 edges. Although they have performed a simple heuristic for resolving aliases, i.e. determining which IP addresses belong to the same router, and have discovered in that way 200 aliases (5 % of the total number of nodes), some apparently different nodes actually represent the same node. Figure 5.8 gives the node degree distribution in the aforementioned graph, plotted on a log-log scale, and fitted with

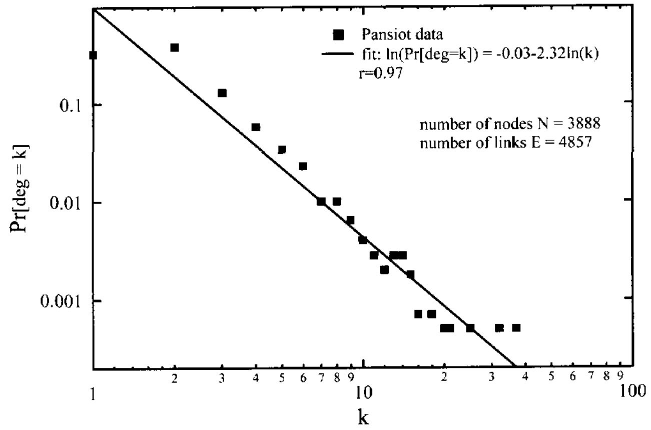


Figure 5.8: Pansiot and Grad (1995).

the linear function decaying with the rate $\alpha_r = 2.32$, with the correlation coefficient[1] $r = 0.97$.

Subsequent to Pansiot and Grad's attempt, several global router-level Internet mapping projects have been initiated, almost all based on the *traceroute* utility [20, 52, 2].

Burch and Cheswick [20] have used BGP backbone routing tables in order to determine the destinations of *traceroutes*. For each prefix in the table, they repeatedly generated a randomly chosen IP address from within that prefix. From *traceroutes* to each such address, they determine router adjacencies, building a router level map in this manner, without applying any alias resolving technique.

Govindan and Tangmunarunkit [52] obtained a snapshot of the Internet topology with the use of the Mercator program. Mercator is designed to map the network from a single source without an initial database of target destination nodes for probing, but to the heuristically determined destination address space. Mercator sends hop-limited probes from a single source and distributes them in directions other than radially from the sender with the use of the source-routing. In this way Mercator discovers crosslinks that might otherwise not have been discovered. Govindan and Tangmunarunkit have collected a large dataset in 1999, resulting in a graph consisting of 228263 nodes and 320149 links.

Regarding the node degree distribution, the results of Govindan and Tangmunarunkit [52] indicated that for node degree values below 30, the plot on a log-log

---

[1] The linear correlation coefficient measures the extent of linear relationship of two variables, and is given by $r = \frac{cov(y,x)}{\sqrt{var(y)var(x)}}$.

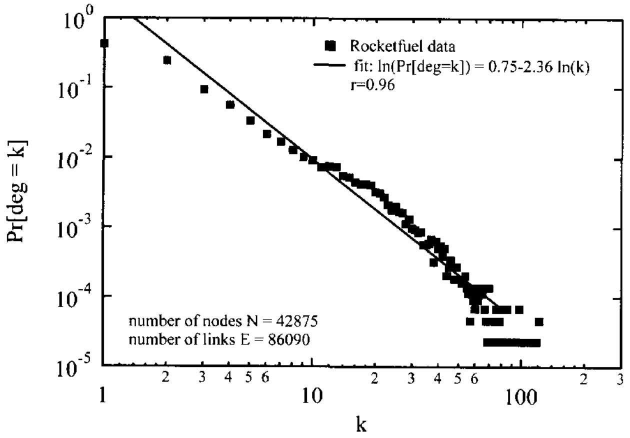


Figure 5.9: Rocketfuel data (January 2002).

scale is linear, suggesting a power-law behavior. However, the distribution becomes significantly more diffuse for node degrees larger than 30.

The Rocketfuel project [94] had the goal to map 10 diverse ISP networks. Publicly available *traceroute* servers served as sources, while the destinations have been chosen out of the BGP routing table, or out of RouteViews when BGP tables were not available. Direct probing has been used, but with a limited number of measurements. *Traceroutes* that contributed most to the map were chosen, and the others were omitted, trading accuracy for efficiency. As discussed in Section 5.1.3, Rocketfuel deploys the most advanced alias resolution technique so far. After aliases have been resolved, Spring *et al.* [94] concluded that almost 70% of the routers had only one interface, while 10% of routers had two aliases, this number rising to one router having up to 24 aliases.

In our analysis, we have used two sets of data obtained within Rocketfuel project [94]. First, we have merged the tables for these 10 ISPs' topologies together, and after eliminating duplicate nodes and links (links and nodes common for several ISPs), we created the aggregate topology with the size of 42875 nodes and 88437 links. The nodal degree distribution in the resulting graph has been calculated and is shown in Figure 5.9.

Furthermore, it is interesting to investigate how the alias resolving process affects the node degree distribution. Accordingly, raw *traceroute* data from the Rocketfuel project, before router IP alias resolution, has also been analyzed, using a few Perl scripts from the Rocketfuel architecture itself. Again, we obtained the aggregate topology over all ISP's, counting 48399 nodes and 103681 links. The degree distribution is plotted in

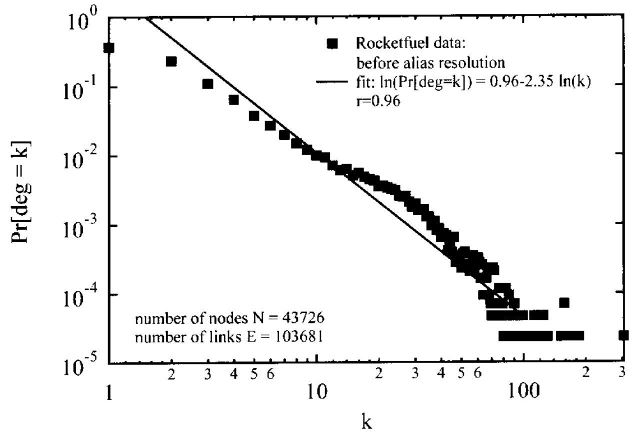


Figure 5.10: Rocketfuel data (before alias resolution procedure, January 2002).

Figure 5.10.

Figures 5.9 and 5.10 seem to indicate that alias resolution process does not influence the node degree distribution significantly. Both distributions follow a power-law, with a slightly different exponent, 2.35 and 2.36 for before and after alias resolving respectively.

5.3.2 Node degree distributions based on CAIDA, RIPE NCC and PlanetLab data

Additionally, we have created maps of (part of) the Internet using CAIDA, RIPE and PlanetLab measurement data.

By merging all *traceroutes* collected from CAIDA as described in 5.1, a router-level map is obtained, from which the node degree distribution is computed, and illustrated in Figure 5.11. The quality of the fit on the log-log scale implies a power-law. Important to note is the higher value of the slope coefficient $\alpha_\tau = 2.97$.

Similarly, a map of the Internet has been constructed from RIPE measurement data, by assembling together the most dominant non-erroneous *traceroute* paths from each of x test boxes (x varied from 31 in dataset 1 to 70 in dataset 3) to all the other boxes in the period 1998-2001 (dataset 1), May-June 2003 (dataset 2), and January-February 2004 (dataset 3). In this way the graph named G_1 has been created, consisting of 1888 nodes and 2628 links, 2574 nodes and 3922 links, and 3850 nodes and 6743 links, in three different periods respectively. To verify if the measured fragment is sufficiently dense, for the graph G_1 created in 2001 we observed the following: if it was constructed

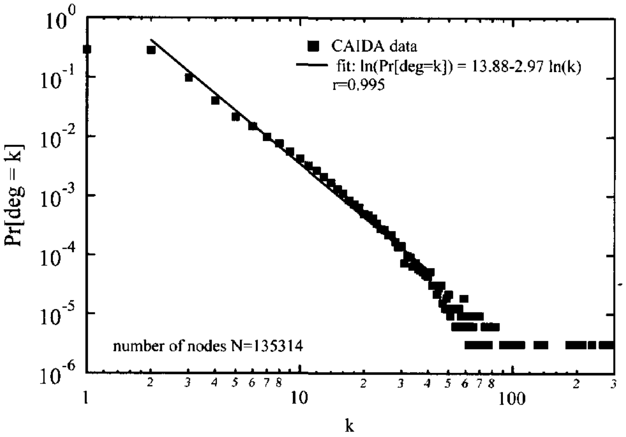


Figure 5.11: CAIDA data (1,2,3 April 2003).

by merging paths collected from (any) 16 sources to all destinations, nearly 85% of the total number of nodes would already be included in the graph. Moreover, for any given source from 31 sources, we compute how many new nodes and links are added to the topology when adding all of the most dominant *traceroutes* collected at that source. It appears that on average only 1.5% (±1%) new nodes, and 0.7% (±0.5%) new links are being added to the topology. We should point out that no effort has been made to determine the aliases, hence, the graph G_1 represents the approximation of the Internet interface map, not of the Internet router map. In Figure 5.12 the probability density function (pdf) of the node degree in the graph G_1 created from data out dataset 1 has been plotted on a log-lin scale. We observed then that the best fit for the pdf obtained was on a log-lin scale. The pdf of the node degrees in G_1 followed an exponentially decreasing function with rate 0.668 over nearly the entire range. This decay rate is very similar to the decay rate of the pdf of the node degrees in the URT, which equals $\log 2 = 0.693$ (see Equation (2.10) in Section 2.3). This is a quite intriguing result, since all the other published results based on many different measured datasets indicate that the degree distribution of the (sub)graph of the Internet should obey a power-law. Moreover, it is not difficult to see that, in general, the union of two or more trees (a) is not a tree and (b) has most degrees larger than that appearing in one tree. Hence, the close agreement points to the fact that the intersection of the trees rooted at a measurement box towards the other boxes is small, such that the graph G_1 is "close" to a uniform recursive tree. This conclusion is verified as follows. Starting with the first union $T = T_1 \cup T_2$ of the trees T_1 and T_2, (where a tree T_i is constructed as a

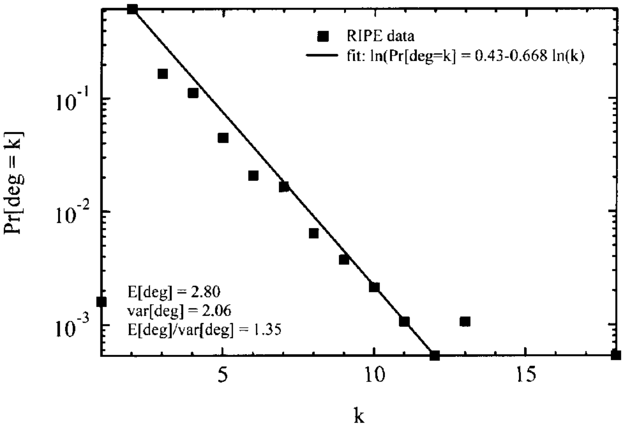


Figure 5.12: The node degree distribution based on RIPE NCC data (dataset 1).

union of *traceroutes* from the source i to all the other measurement boxes), we also have computed the intersection $T_1 \cap T_2$. Subsequently, we added another tree T_3 to the union $T = T_1 \cup T_2 \cup T_3$ and again computed the intersection (or overlap) $T = (T_1 \cup T_2) \cap T_3$. This process was continued until all trees were taken into account. We found that (a) the number of nodes in the intersection was very small and (b) the common nodes predominantly augmented the degree by 1 and in very few cases by more than 1. It is likely that the overlap of trees would be larger (in terms of common nodes and links) if we had considered the router level map instead the interface map. The similarity in properties of the graph G_1 and the Uniform Recursive Tree might be in that case smaller. The Rocketfuel project results have indicated that there is a small impact of alias resolving to the node degree in Internet maps. Hence, we don't expect that alias resolution only can explain the discrepancy.

We performed the same analysis for the dataset 2 and dataset 3, with more testboxes in RIPE architecture. The results obtained were slightly different, as illustrated in Figure 5.13. The slope coefficient decayed to the value of 0.61 and 0.41 for dataset 2 and dataset 3, respectively. Even though the slope coefficient does not correspond to that of a URT as observed in 2002, the data is still better fitted with a linear function on a log-lin scale, than on a log-log scale. We anticipate that one of the possible reasons may lie in the fact that each RIPE measurement box acts as source as well as destination.

Finally, the data from the PlanetLab network has been used. By merging the *traceroutes* from each of 79 nodes to all the other, the topology consisting of 4226 nodes

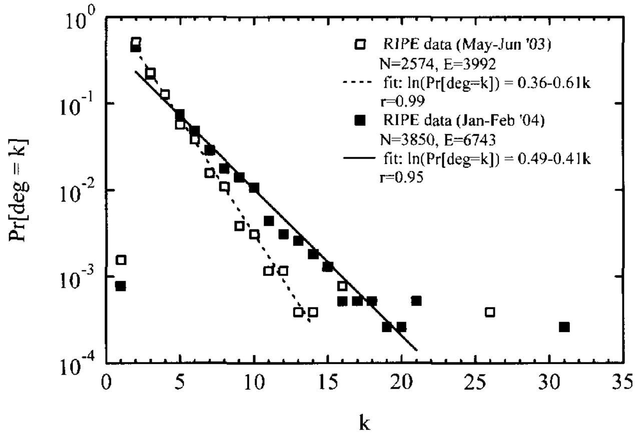


Figure 5.13: The node degree distribution based on RIPE NCC data (dataset 2 and dataset 3).

and 7171 links was produced. No alias resolution technique has been implemented. The node degree distribution in this map has been computed, and illustrated in Figure 5.14.

Again, a higher quality fit has been achieved on a log-lin than on a log-log scale, even though the slope coefficient takes the value 0.48. This result seems to confirm our observation that when each source serves as the destination as well, the node degree distribution observed seems to follow an exponential function.

5.3.3 Simulations of the union of Shortest Path Trees in the complete graph

If we assume that *traceroutes* represent shortest paths, than the subgraph of Internet graph G_1 in Section 5.3.2 can be modelled as a union of shortest paths. This triggered us to investigate the node degree distribution in the union of Shortest Path Trees in a complete graph K_N with uniformly distributed link weights, and to compare them with distributions obtained from RIPE data, for small values of m compared to N. We have performed the following set of simulations: in a complete graph K_N with uniformly distributed link weights consisting of $N = 1000$ nodes, m nodes have been chosen randomly, where $m = 3, 5, 10, 20, 50, 100, 200, 500, 700, 1000$. Uniformly distributed link weights represent the special case of polynomially distributed link weights, $P[w \leq x] = x^{\alpha} 1_{0 \leq x \leq 1} + 1_{x \geq 1}$, with parameter $\alpha = 1$. For a particular m, the Shortest Path Tree rooted in each of m nodes to the other $m - 1$ destinations has been computed. Consequently, the union of the shortest paths has been created, and the node degree

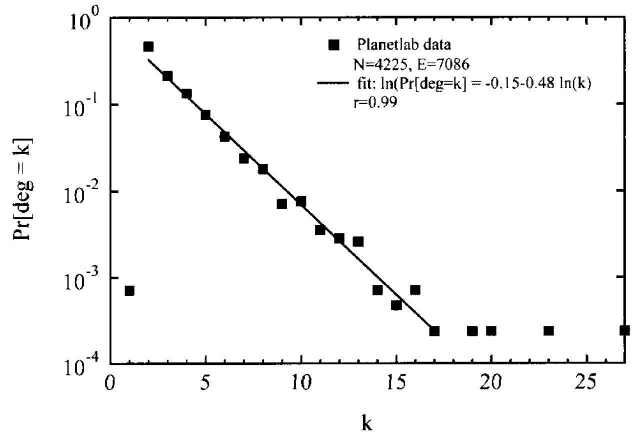


Figure 5.14: The node degree distribution based on PlanetLab data.

distribution in the resulting graph has been calculated. For each m, 10000 iterations have been performed. The same procedure has been applied for every value of m. The resulting distributions are presented in Figure 5.15. Figure 5.15 reveals that for small m compared to N the simulated pdf of node degrees resembles the shape observed from RIPE data, except for several outliers in Figure 5.13. Perhaps tuning the parameter α would fit the distribution given in Figure 5.13 better.

5.4 The Structure of Multicast Trees

5.4.1 Related work

To the best of our knowledge, only few results have been published on the characteristics of multicast routing trees. The first one has been provided by Chalmers and Almeroth [25], who have looked into the properties of the Internet multicast trees on the MBone [43]. They have gathered multicast tree data for four live multicast sessions: the 43rd IETF meeting in December 1998 and the NASA shuttle launch in February 1999, each of them consisting of a separate audio and video channel. The path from each receiver to the source has been traced via mtrace (multicast *traceroute*) [49]. However, since receivers are traced one after another, the receivers participating for a short time may have been missed. Indeed, only 43% of the receivers for IETF and 29% for NASA have been successfully traced. The mtrace data has been used for each dataset to reconstruct a multicast tree. Chalmers and Almeroth have developed the tool mwalk, that builds

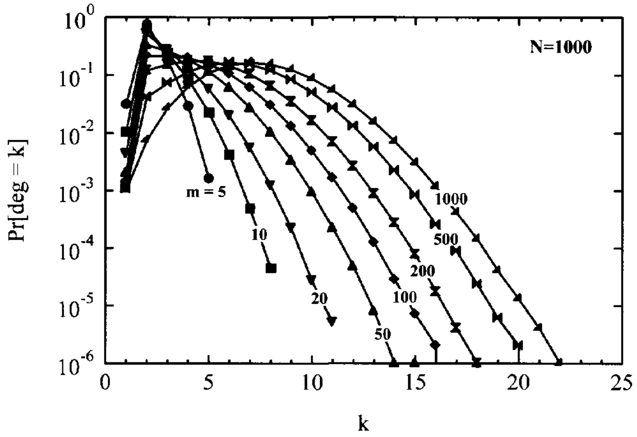


Figure 5.15: The node degree distribution in the union of shortest path trees in K_N ($N = 1000$).

an activity graph of all possible trees over time: the whole session has been divided in 10000 intervals, and to each receiver an activity table is assigned, with the time intervals in which that particular receiver was a member of the group. In Figure 5.16 we have plotted the node degree distribution obtained for 10000 realizations of the multicast tree, in the 43rd IETF video dataset, when 129 receivers have been traced to belong to the group.

After fitting their data on different scales, we noticed that the best fit of all (with the correlation coefficient of 0.91) was obtained for the linear fit on the log-lin scale, suggesting rather exponentially than polynomially distributed node degrees. Moreover, the slope of the curve in Figure 5.16 is approximately the same as that of the Uniform Recursive Tree (URT) (see Section 2.3). However, it must be noted that the data-set is rather small and, therefore, we must be careful in drawing hard conclusions.

The only other result on the multicast tree degree distribution so far has been provided by Dolev *et al.* [39]. In [39] Dolev *et al.* have investigated properties of multicast trees obtained from Internet measurement data. The data they used for their multicast analysis has been obtained via unicast *traceroute* measurement. They have used two datasets: first, on the underlying topology provided from a mapping project presented in [20] (using *traceroute* measurements), they generated Shortest Path Trees using the Dijkstra algorithm. The second dataset was created based on *traceroute* measurements of the paths between the root and the clients in the client population of www.bell-labs.com. Dolev *et al.* do not report whether loops occurred in their data, nor how they approached and treated that phenomenon. In [39, Figure 6 and Figure 7]

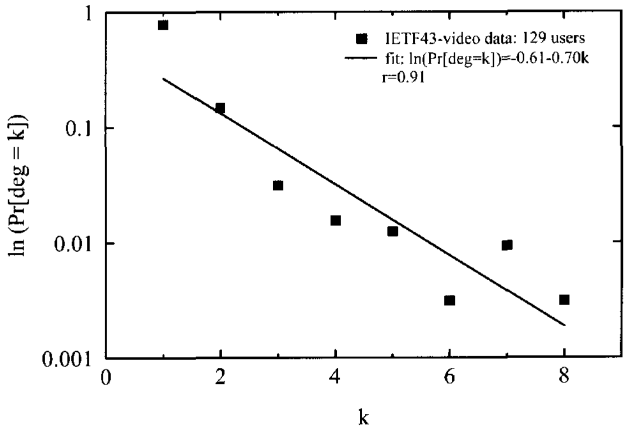


Figure 5.16: Node degree distribution in IETF 43rd meeting-video data set (7-11 December 1998) from [25].

they have plotted the node degree distributions in both datasets on a log-log scale, and fitted with the linear function decaying with the rate −3.40 and −3.18 for the first and the second dataset, with the correlation coefficients of 0.9897 and 0.9829, respectively. These findings seem to suggest a power-law structure of the node degree distribution in multicast routing trees, which contradicts that of Figure 5.16.

5.4.2 Node-degree distribution of trees in RIPE and CAIDA

In order to understand the discrepancy found in Section 5.4.1, we have further investigated the node degree distribution obtained from the Skitter project. The node degree distribution for each of the trees has been computed. We present the resulting distributions only for trees rooted at the source in the United States, since the distributions for trees rooted at two other sources are almost identical. Our results indicate that based on CAIDA *traceroute* data, for $m \geq 100$, as given in Figure 5.17 ($m = 1000$ and $m = 100$), the node degrees seem to be polynomially distributed.

However, when we plotted the node degree distribution for $m = 50$ on two different scales, log-lin in Figure 5.18(a), and log-log scale in Figure 5.18(b), we noticed one remarkable property: when fitting the data with a linear function in both scales, the quality of the fit seems to be comparable. This can be seen in the value of the linear correlation coefficients r_α and r_β, that represent the measure of the quality of fit. The resulting slope coefficient α_τ and the correlation coefficient r_α for linear fits on log-log scale, and the slope coefficient β_τ and the correlation coefficient r_β for linear fits

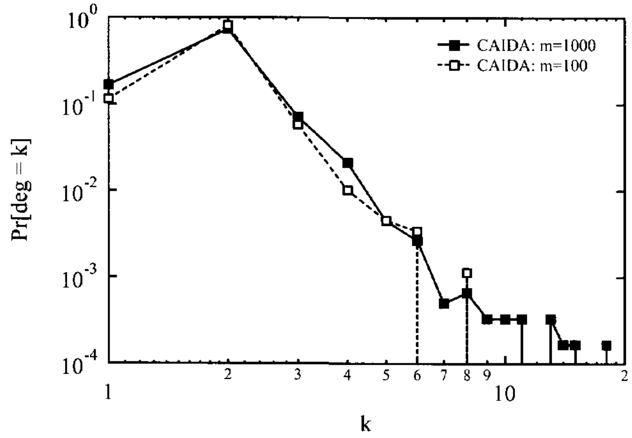


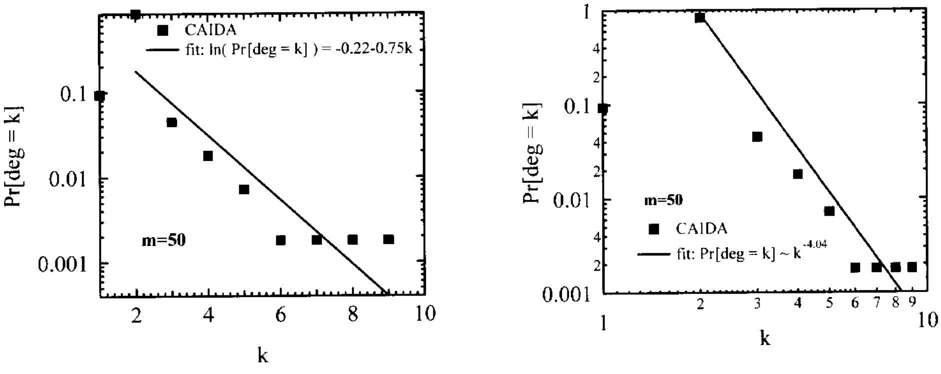
Figure 5.17: The degree distribution in multicast tree based on CAIDA traceroute data ($m = 1000$ and $m = 100$).

$m = 50$	
$\alpha_{\mathcal{T}}$	4.04
r_α	0.95
$\beta_{\mathcal{T}}$	0.75
r_β	0.93

Table 5.2: Linear function fit coefficients (log-log scale and log-lin scale), with corresponding Pearson's coefficient in CAIDA data for m=50.

on log-lin scale, for $m = 50$ destinations derived from CAIDA *traceroute* data, are summarized in Table 5.2. We notice that the quality of the fit on a log-log scale for $m = 50$ is only slightly higher than the quality of the fit on the log-lin scale. Therefore, it is questionable whether based on the given data it is reasonable to claim that the degree distribution of the union of *traceroutes* representing a multicast routing tree follows a power-law distribution for small m.

Even more doubt is raised after plotting the data obtained from RIPE in two different scales, for $m = 20$ and $m = 50$ (Figure 5.19). Although fitting the data on a log-log scale is somewhat better, the quality of fitting with linear functions in both scales is considerably deteriorated, as can be seen from Table 5.3 and Figure 5.19. Therefore, we conclude that the degree distribution in the multicast tree for small m does not convincingly follow a power-law. For larger values of the number of destinations m the degree distribution seems to follow a power-law, when multicast trees are created as a



(a) (b)

Figure 5.18: (a) The degree distribution in multicast tree based on CAIDA traceroute data ($m = 50$) on log-lin scale. (b) The degree distribution in multicast tree based on CAIDA traceroute data ($m = 50$) on log-log scale.

$m = 50$		$m = 20$	
$\mathbf{r}_\alpha$	0.78	$\mathbf{r}_\alpha$	0.74
r_β	0.61	r_β	0.64

Table 5.3: Pearson's coefficients for linear function fit in log-log and log-lin scale in RIPE data for m=50 and m=20

union of *traceroutes*. The result of Chalmers and Almeroth implies that if a method for constructing trees is used other then the union of *traceroute* paths, power-laws might not be observed for even larger values of m. Nevertheless, as in the case of the result of Chalmers and Almeroth presented above, also for the results based on RIPE and Caida measurements holds that one must be careful when drawing hard conclusions based on these limited data sets only.

5.4.3 Number of links in CAIDA trees-the gain of multicast

In Figure 5.20 the number of links has been plotted, for each source, as a function of m. In addition, for $N = 135314$ (the number of nodes in the Internet map derived from *traceroute* measurements) and for various values of m (in the range $[50, 20000]$), the values of the functions $E\left[H_N\left(m\right)\right]$ and $E\left[H_N\left(m\right)\right] \pm 3\sigma_N(m)$ (where $\sigma_N(m) = \sqrt{var\left(H_N\left(m\right)\right)}$) have been computed, according to (2.7) and (2.8), and plotted in the same figure. We observe that the measured number of links falls in the range $E\left[H_N\left(m\right)\right] \pm 3\sigma_N(m)$, indicating again the URT nature of the multicast trees. Further, since $\sigma_N\left(m\right)$ is much smaller than $E\left[H_N\left(m\right)\right]$, the number of links in the URT is well approximated by the mean, $H_N\left(m\right) \approx E\left[H_N\left(m\right)\right]$ for large values of N. Thus, the measurement results so far indicate that the URT represents a reasonable, first order model for the multicast tree in Internet.

5.5 Conclusions

Several ambiguities related to *traceroute* data have been discussed, such as errors and inaccuracies, alias resolving and bias sampling, that make the trustworthiness of derived conclusions on the topological characteristics of Internet questionable. We have ascertained that 17% of the *traceroutes* we collected in 2001 were erroneous (loops, etc.). The analyzed routing pathology is summarized and compared to results obtained by Paxson in 1995. We may conclude that the quality of *traceroute* measurements either decreased over time or that the part of the Internet covered in Paxson's measurements is less prone to error than that in the RIPE measurements.

Further, the detailed review of Internet map node degree distribution has been provided. In Table 5.4 we summarize the presented results. By x we denoted the data that was not available to us. Most of the measurements, with the exception of the RIPE and PlanetLab data, indicate power-laws, with similar values for the gradient coefficient α_τ. Furthermore, they seem to suggest that alias-resolving techniques do not have a major effect on the node degree distribution in the Internet. The gradient coefficient α_τ lies in the range $2.3 - 2.4$, with the exception of CAIDA data, where the value of the slope coefficient is $\alpha_\tau = 2.97$. This might be the consequence of the one order of magnitude larger map obtained with Skitter, than the one obtained within the

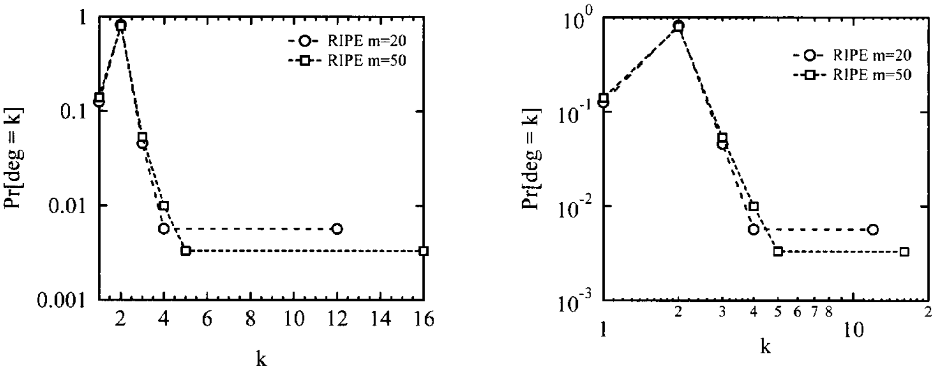
data set	number of trace-routes	number of nodes	number of links	slope coefficient (log-log scale)
Pansiot and Grad	46110	3888	4857	2.3237
CAIDA	x	276680	324338	2.9723
Rocketfuel before alias resolving	1048808	43726	103681	2.3424
Rocketfuel after alias resolving	1048808	42875	86090	2.3683
Mercator	x	228263	320149	2.38

Table 5.4: Comparison of node degree distribution from various traceroute data sets: power-law.

Rocketfuel project.

The divergency in results obtained by RIPE NCC and PlanetLab may be caused by the RIPE and PlanetLab measurement architecture, where each measurement box serves both as the source and as the destination. Hence, the number of sources and destinations is balanced, whereas in all the other measurement architectures the number of sources has been very limited (only few) compared to a large number of destinations. The discrepancy in the node degree distributions based on RIPE and PlanetLab *traceroute* data with the previous results seems to confirm the above stated observation on the reliability of *traceroutes*.

Finally, we have analyzed the structure of multicast trees, created as the union of *traceroutes*. The scarce results on node degree distributions in multicast trees have been controversial as well. Our results on the node degree distribution in multicast trees seem to suggest that based on *traceroute* data, universal power-law behavior cannot be claimed. While for larger values of the number of destinations m the degree distribution seems to follow power-law, for $m \leq 50$ based on both RIPE and CAIDA *traceroute* data this seems not to be the case. Furthermore, the Internet measurements of the number of links in the tree seem to suggest that the URT model we used to derive the laws given above represents a reasonable model for multicast trees in Internet.



(a) (b)

Figure 5.19: (a) The degree distribution in multicast tree based on RIPE traceroute data ($m = 50$ and $m = 20$) on log-lin scale. (b) The degree distribution in multicast tree based on RIPE traceroute data ($m = 50$ and $m = 20$) on log-log scale.

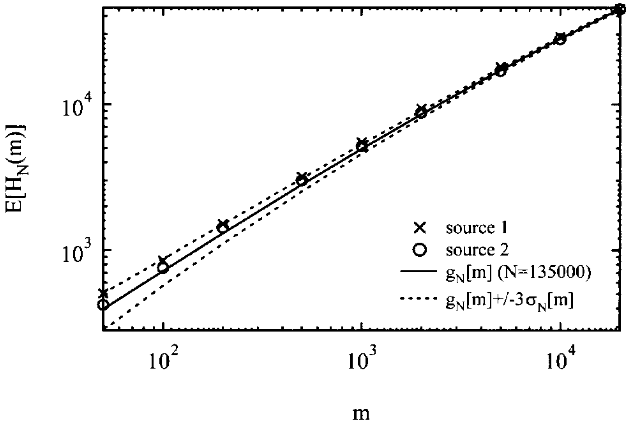


Figure 5.20: The average number of links (Caida measurements and theoretical value).

Part II

Multicast on the Application Layer

Chapter 6

Overview of AL Multicast Protocols

Although relatively young, the idea of application layer multicast has attracted much research attention, resulting in a number of proposals. In Application Layer multicast, as the name implicates, the multicast related features, such as group membership, packet replication and multicast routing are implemented at the end hosts, rather than in network routers, as depicted in Figure 1.4. Evidently, Application Layer multicast suffers from worse efficiency than network layer multicast, since packets may traverse the same link several times. Moreover, end hosts in general do not possess the routing information available to routers, instead they must rely on end-to-end measurements to infer network metrics upon which the end-host multicast delivery tree is built.

All application layer multicast protocols organize the group members into two topologies, namely the control topology and the data topology. Members that are peers in the control topology exchange periodic refresh messages to identify and recover from "ungraceful" departures from the group[1]. The data topology is usually a subset of the control topology and identifies the data path for a multicasted packet on the overlay. In fact, the data topology is a tree, while the control topology ensures a greater connectivity between members. For this reason, in many protocols the control topology is called a mesh and the data topology is called a tree.

The proposed application layer multicast methods can be classified in various ways. One criterion is how the overlay nodes are selected. According to this criterion, the protocols can be classified in the fixed-nodes-based and dynamic-nodes-based approaches. In the fixed-nodes-based approach [27, 9, 28], nodes that form the overlay multicast trees are first strategically placed across the whole Internet. Their advantage is that since fixed nodes are used, the multicast tree is stable. However, the multicast service is not flexible, and still needs the ISP support. In addition, those fixed nodes can possibly form a bottleneck. In dynamic-nodes-based overlay multicast [50, 80, 56, 109, 88, 71, 104, 11, 16, 98] the group members are self-organized into

---

[1] An ungraceful departure is one when the member departs from the group without informing its peers through control messages.

an overlay multicast tree and periodically self-improve. Another criterion for protocol classification is whether the overlay building algorithm is centralized or distributed.

Here we adopt the approach to classify the AL multicast protocols based on the type of topology created for communication between end hosts. According to this criterion, they could be roughly classified into four different categories: mesh-based, tree-based, multicast on top of structured peer-to-peer overlays, and hierarchical multicast. In this chapter, we will explain in more detail each of these categories.

6.1 Mesh-based (mesh-first) multicast

In the mesh-based approach [28, 31][31] group members first self-organize into the mesh topology, so that between each pair of members multiple paths exist. The second step is the creation of a source-specific tree rooted at any member, achieved by running some of the well known multicast routing mechanisms, e.g. DVMRP [103]. We will describe Narada [31], the most representative protocol belonging to this class and Scattercast [27].

6.1.1 Narada

How it works: Each member in the group, i.e. each node in the mesh, maintains the list with the state of all the other members in that particular group. A new member that wants to join this group needs first to obtain the list of current members. It receives this list from the node designated as a Rendezvous Point, with the help of a bootstrap protocol. Subsequently, the new member sends the join request message to several members from the list, selected randomly, and tries to join the group as their neighbor. The member has successfully joined when it receives the acceptance message from at least one of the queried nodes. The new member will then start exchanging periodic refresh messages with its neighbors on the mesh, updating their tables. This information will be further propagated, until it reaches all members of the group.

If the member wants to leave, it will send the leave message to the neighbors on the mesh. Also this information will be propagated to all the other nodes in the mesh.

The multicast data delivery tree in Narada are the source-based trees, with any specific member as the source. The delivery tree is computed by the members of the group, which run a variant of a distance vector protocol. The tree is then constructed from reverse shortest paths between each recipient and the source, in identical fashion to e.g. DVMRP [103] (DVMRP has been explained in Chapter 3). An example of such a data tree is shown in Figure 6.1(a).

One of the main design objectives of the Narada creators was robustness. In case of a failure, for instance of node 2, upon not receiving any refresh messages from this user, nodes 1, 3, 7 and 8 will query node 2. If they receive no reply, node 2 will be

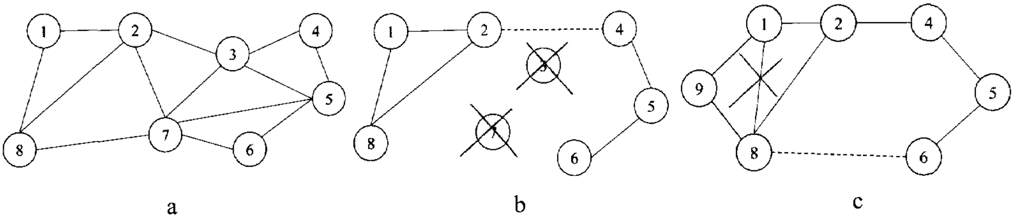


Figure 6.1: Narada protocol.

considered "dead" and the rest of the members will be informed of it. A more serious consequence would be induced by the failure of the nodes 3 and 7, leading to a mesh partition 6.1(b). Members will detect the partition by the absence of the updates from members belonging to the other cluster. Each member keeps the list of members that it receives no updates from. Each entry in that list can remain in the list a limited amount of time. Periodically and with certain probability, each node that detected partition chooses and probes one of the nodes in that list. If it receives a response, it will try to add the link to that node and repair the partition. Otherwise, the probed node will be regarded as dead. The value of this probability is chosen carefully so that in spite of several members simultaneously attempting to repair a partition only a small number of new links are added.

Due to several possible causes, the constructed mesh might not be optimal: (i) initial selection of neighbors by a joining node is random, (ii) links created during the partition reparation process are necessary at that moment, but eventually become superfluous; (iii) dynamic join/leave process; and (iv) dynamic conditions of the underlying network. Since the data delivery paths in Narada are spanning trees of the mesh, their quality depends on the quality of links that constitute the mesh. Therefore, Narada allows periodical refinement of the mesh by adding and dropping of overlay links. Each member periodically evaluates the cost and the utility of the existing links, as well the cost and utility of forming a link to some random member that is not a neighbor yet. An example is given in Figure 6.1(c). A joining node 9 (Figure 6.1(c)) chooses randomly two nodes to connect to, e.g. node 1 and node 8. In Figure 6.1(c) a link evaluated as potentially useful (the link between nodes 6 and 8, reducing the number of hops on the underlying topology between e.g. nodes 6 and 9) is included. A non-useful (redundant in this example) link (between nodes 1 and 8) will be omitted from the mesh.

6.1.2 ScatterCast

ScatterCast [27] belongs to the group of the fixed-nodes based protocols: its architecture is composed of dedicated ScatterCast Proxies (SCXs) strategically placed in service clusters around the Internet, usually at ISP points of presence.

How it works: SCXs use a protocol called Gossamer to distributedly self-organize in a mesh structure. For each multicast session a new mesh is generated. Members that wish to join first contact a nearest service cluster, and request an SCX. The cluster sets up an SCX if there is none, and communicates its IP address to the new member. In addition, it sends its locally scoped multicast address, so that new members can communicate with SCX either via unicast, or local area multicast if possible. Similar to Narada, a distance vector protocol is run by SCXs on top of this mesh to construct a source-rooted delivery tree. In the tree creation process, two requirements are considered: the degree of SCX is limited according to its bandwidth and the delay between the source and the destination SCXs is minimized.

Periodically, in order to improve the mesh, SCXs probe each other, and evaluate the cost of potential links compared to the cost of existing links. The cost of a link is computed as the sum of the costs of paths to all source SCXs. If the new cost is lower than the cost of the existing link, the existing link will be substituted with the new one.

Mesh partitions are detected and repaired with the use of centralized Rendezvous Points. One of Rendezvous Points is randomly selected to periodically send refresh messages along the mesh. If any of the SCX nodes stops receiving these messages, it contacts that particular RP and reconnects the mesh. Although the partition recovery problem is considerably simplified with the use of centralized points, it will still arise in case all Rendezvous Points fail.

6.2 Tree-based (tree-first) multicast

Protocols belonging to this group distributedly construct a data delivery tree directly. Moreover, in contrast to mesh-based protocols, group-shared trees are built. The nodes organize themselves by choosing their parents in the tree. The choice of parents is crucial for the performance and the efficiency of the tree. In order to enhance the overlay, some of the protocols in this group build a mesh subsequently. Members of the group connected in a tree discover some other members that are not their neighbors on the tree, and set up the additional control links to these members. The protocols of this class differ among each other in the tree building algorithm.

6.2.1 Banana Tree Protocol (BTP)

The Banana Tree Protocol (BTP) [56] is probably the simplest protocol belonging to the class of tree-based protocols.

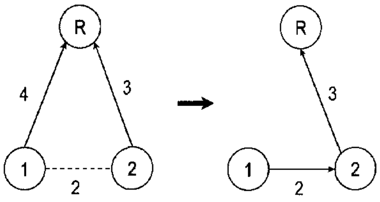


Figure 6.2: Tree optimization in BTP: Parents switching.

How it works: A node that wants to join the group obtains the information on some nodes already existing in a group with the use of a bootstrap mechanism, and chooses one of them to be its parent. If the node is the first in the group, then it becomes the root. The tree is incrementally optimized with the occasional parent switching. The node changes its parent in order to reduce tree cost or latency. It can switch to its nearby sibling or grandparent, but not to any other arbitrary node, such as its descendants, to prevent the creation of loops. Each node receives the list of siblings from its parent, and by pinging its siblings looks for a sibling closer than its parent. In case such a sibling is found, a node sends a switch request to that sibling (Figure 6.2). At the same time, the node changing the parent prohibits any other sibling to switch to it, otherwise loops and partitions may be formed, as depicted in Figure 6.3(a). In addition, it has to ensure that the node it is switching to is still its sibling, to avoid the situation outlined in Figure 6.3(b). Nodes 1 and 3 are in the first instance the siblings of node 2. However, due to their change of parents, this is not the case any longer at the moment node 2 decides to change its parent. Hence, at the moment of switching, the switching node must possess the current parent information.

6.2.2 Host Multicast Tree Protocol (HMTP)

In the Host Multicast Tree Protocol (HMTP) [109] a group-shared tree is built. The tree is built in such a way that the cost is minimized, and the maximum degree at each node is sustained. The optimization metric is a member-to-member round-trip time (rtt). Zhang *et al.* suggest in [109] to add bandwidth as a second metric in the future. HMTP does not create a control mesh.

How it works: The join procedure is described in Figure 6.4. A member that wants to join (node 8) contacts the root (R), and sets it as a potential parent. Next, from the root it obtains the list of the root's children nodes (nodes 1, 2 and 3). Based on the *round-trip-time* values, it finds the node closest to itself among the potential parent and its children. If the closest node is the potential parent, the newcomer sends the

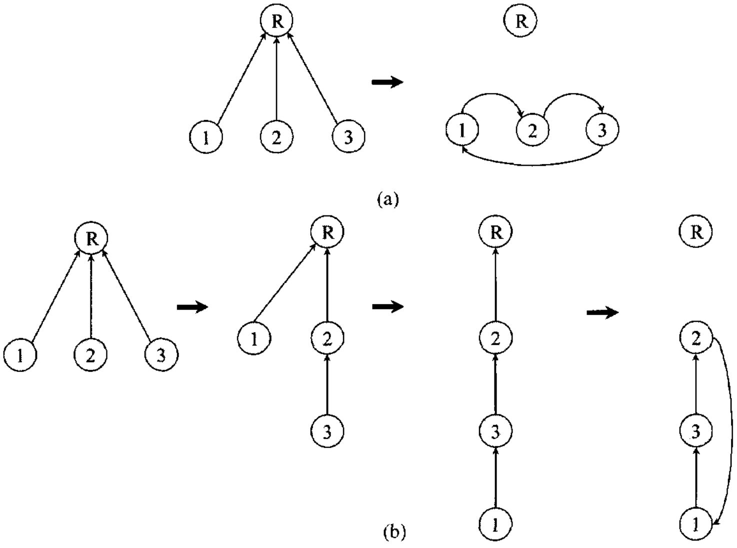


Figure 6.3: (a) Loop formation in BTP: simultaneous switching (b) Loop formation in BTP: outdated information.

join request to that node (e.g. R). A potential parent accepts a newcomer as its child, provided the number of its children is smaller than its degree bound. Otherwise, the nearest of potential parent's children becomes the potential parent (e.g. node 3), and the whole process is repeated all over: the newcomer obtains the list of children of the potential parent (only node 4 in our example), and searches for the nearest node among potential parent and its children. In this way, the nodes that are nearby on underlying topology will be clustered together.

The tree structure is maintained by periodical exchange of refresh messages among the neighbors in the tree. Each member keeps information on its current children, and on its path to the root. The path information is included in the refresh messages of parents toward their children. When a node wants to leave, it informs its parent and children. The parent of the leaving node then simply deletes that node from its children list. The children of the leaving node have to look for a new parent. The search for a new parent is identical to the join procedure, except that the order is reverse: instead by the root, the children of the leaving node start the procedure at their grandparent.

The partition recovery procedure is similar to the member-leave procedure. Absence of the refresh messages is the indication to the parents and children of the failed node that a failure, hence partition, has occurred. Upon failure detection, the parent of the failed node updates its children list. Children reconnect to the tree by trying to connect

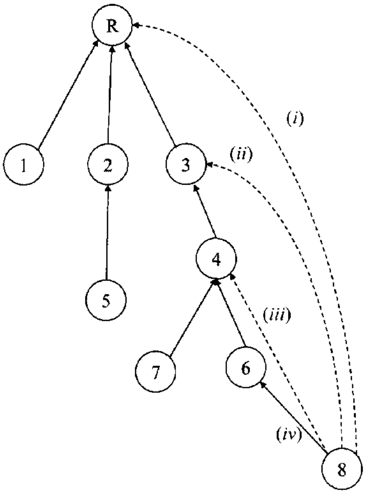


Figure 6.4: HMTP join.

to some node on their root paths.

The dynamics of the underlying network and of the group membership can decrease the quality of the tree over time. The tree can be refined by members changing the parent. A member occasionally starts a rejoin procedure, not from the root, but from another randomly selected node in its root path. In addition, in order to search other branches for a potentially better parent, a node may choose a suboptimal node for a parent (not the one closest to himself). If a better (i.e. closer) parent is found, the node will switch to that parent.

In HMTP no loop avoidance algorithm is applied. Instead, loop detection and resolution is performed. A member in the loop detects the loop by finding himself in its root path. When the loop is detected, the member in the loop leaves the current parent, and rejoins the tree, searching for a parent from the root on.

6.2.3 Yoid

Your Own Internet Distribution [50] was probably the first protocol in the group of tree-based protocols. Yoid builds a group-shared tree as well, along which data is disseminated among the members. However, to increase robustness and stability, Yoid also builds control mesh topology. These additional mesh links are used for recovery from tree partitions.

How it works: When a new host wants to join a group, it contacts a rendezvous point, which is not necessarily a member of the group. RP provides a newcomer with a list of (several) current, randomly selected members. In case the joining node is the first member of the group, it becomes the root. Otherwise, it selects one of the members in the list as a parent.

This simple tree building mechanism might lead to a structure that suffers from loops and bad performance. To prevent loops, Yoid deploys a simple loop avoidance mechanism: a node rejects join requests from nodes that are in its root list. This rule prevents most of the loops, but not all of them. If the loop occurs, it is detected in the same manner as in HMTP: a node finds himself in the root path obtained from its parent. To avoid transient tree reconfiguration when several nodes try to reconnect to new parents simultaneously, Francis *et al.* [50] propose additional information in the root path information, coined *switchstamp*. Each time a node changes the parent, in the first root path information message it receives from a new parent, the *switchstamp* of that node is set higher than that of any other node in the root path. In case of a loop, a node in the loop with the highest *switchstamp* will terminate the loop by changing the parent.

Another consequence of the simple tree building protocol can be high data loss and high latency. Therefore, two tree refinement mechanisms, latency and loss-rate refinement, are proposed. Whereas the latency refinement algorithm tries to improve the end-to-end latency by effectively reducing the tree depth, the loss-rate refinement algorithm restricts the maximum node degree of the nodes behind low bandwidth links. These two processes run simultaneously so that, despite their conflicting objectives, they form the tree in such a way that both metrics are considered

6.2.4 ALMI

ALMI [80] protocol is probably one of the very first application layer multicast proposals. It belongs to the group of centralized approaches. Due to its centralized character, ALMI is only suitable for groups of small size (several tens) and many-to-many type of applications.

How it works: In ALMI, a session (group) consists of the session (group) controller and regular session (group) members. The session controller is in charge of computing a minimum spanning tree (MST) spanning all members of the group. This tree is used for the dissemination of the application data. As an optimization metric, the latency between the members is used. Each member monitors the latency to a set of other members. The controller receives the updates from the members containing this latency data, and based on this data, it periodically recomputes the tree. Subsequently, the controller communicates to each member the information on the new parent and children nodes. Members willing to join a multicast group first contact and query the controller. The controller returns the identity of a parent node, to which the new member should initiate a connection, and list of children nodes, from which the member should accept connection requests.

To enhance the robustness of the protocol, Pendarakis *et al.* propose in [80] several back-up controllers that operate in stand-by mode. Nevertheless, the scalability of the protocol remains highly restricted.

The centralized approach considerably simplifies the routing challenge, since the centrally computed tree should contain no loops. Loops and partitions might however occur. One reason for their occurrence are the delayed or lost updates from members. Another cause might be the dissemination of the different versions of the MST to members. This problem can be circumvented by assigning a number to each version of the MST. Members maintain a cache of the different version numbers in the routing tables and only route packets with the tree version numbers contained in the cache. If a packet with a newer tree version is received, the receiving node will re-register with the controller to receive the new MST.

6.2.5 Overcast

How it works: Overcast [62] is designed as a single-source multicast scheme, with bandwidth and not latency as a metric for optimization, since is not intended for interactive applications. The objective of the tree building mechanism is to maximize bandwidth from all nodes to the root. The protocol places a new member in an iterative process as far from the root as possible without reducing the bandwidth. A member that wants to join starts the join process by contacting the root, that becomes the potential parent node. In each iteration of the process a member estimates the direct bandwidth to the potential parent as well as the bandwidth to the potential parent via each of the potential parent's children. The bandwidth is estimated by measuring the download time of 10 Kb of data. If bandwidth via any of the potential parent's children is higher than or equal to direct bandwidth to the root, then that particular child node becomes the potential parent node, and the new iterations begins. Otherwise, the potential parent becomes the parent of a new member. When there are several children of the potential parent that satisfy the bandwidth criterion, the child closest (in terms of network hops) to the new node (as reported by *traceroute*) is selected as potential parent.

Occasionally, members measure the bandwidth to their siblings, parent and also grandparent, and switch parent based on the evaluation result.

Each member periodically sends refreshment messages to its parent. If a parent is not contacted by a child for a certain period, the parent considers the child and all its descendants as "dead".

6.2.6 Overlay Tree Building Control Protocol (TBCP)

TBCP [73] builds trees that are constrained in the node out-degree: each member fixes the number of children it is willing to support.

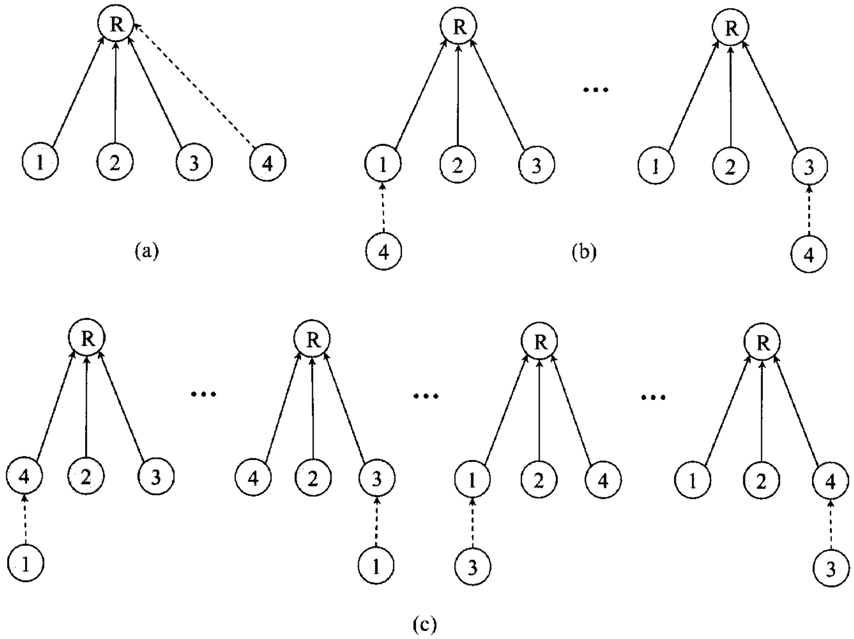


Figure 6.5: TBCP joining mechanism: local configuration evaluation.

How it works: Figure 6.5 demonstrates the join procedure in the TBCP protocol. The node who is the main sender to the group is chosen to be the root. Similar to HMTP and Overcast, the new member (node 4) first contacts the root (R), and receives a list of its children (nodes 1, 2 and 3). The newcomer then measures the distance (rtt) to the root and its children. Subsequently, it returns the measured data to the root. The root then evaluates all possible configurations for the newcomer, as illustrated in Figure 6.5, upon which it selects the optimal one. If a new node is accepted as a child of the root (Figure 6.5(a)), the joining procedure is completed. If, on the other hand, either newcomer 4 (Figure 6.5(b)) or any of the current root's children (nodes 1 to 3) has to be redirected (Figure 6.5(c)), that particular node restarts the joining procedure, but from the node to which it is redirected to.

Also the existing members can occasionally perform the join procedure to replace themselves in the tree, in order to improve the performance of the tree. The tree performance can further be improved by organizing members hierarchically in domains, where each domain has its own root node.

No node failure recovery mechanism is deployed in this protocol.

6.2.7 Short general comparison of mesh-based and tree-based protocols

We conclude this section by briefly summarizing the advantages and disadvantages of both mesh-based and tree-based group of protocols, with respect to each other.

The first advantage of the mesh-based protocols is their superior robustness. The probability of overlay network partitioning due to node failure or departure is low, since these overlays consist of multiple or redundant connections between group members. In addition, there is no need to reconstruct a path (as is the case in a tree overlay) since alternate paths already exist. Another advantage is that the mesh-based protocols utilize the already existing tree building algorithms. Finally, since source-specific trees are built in the mesh, protocols of this group support multi-source (many-to-many) applications.

The main drawback of mesh-based approach is high control overhead, since every member needs to keep information on state of all the other members. Every time member join/leave occurs, all the group members have to be notified, leading to considerable control overhead. Although this was an explicit design choice, trading overhead for greater robustness, it restricts these algorithms to support only small multicast groups. The mesh-based mechanisms also suffer from higher packet loss, as they have to run a tree building algorithm. That can lead to transient conditions when not all the routing tables of the members have converged, and, consequently, packets may be lost.

The main advantage of tree-based protocols is that they scale better to larger group sizes than mesh-based protocols, since each host has to be aware of only a small number of other hosts. Their main weakness is the partitioning susceptibility. Tree overlays are more susceptible to partitioning, since failure (or a voluntary leave) of any (non-leaf) member can lead to the tree partitioning and communication breakage.

6.3 Multicast on top of structured P2P overlays

In the last few years, file sharing systems gained a lot of popularity. Peer-to-peer systems are attractive in at least two respects. First, from the user standpoint, peer-to-peer computing has a huge potential, as it reduces the need for expensive back-end servers, typically used to perform complex tasks. The current trend in building P2P systems consists in providing an application independent overlay network as a substrate on top of which novel large-scale applications can be constructed.

Two broad classes of overlay networks infrastructure solutions can be differentiated: unstructured [51, 93], and structured overlay networks. In unstructured overlays nodes are organized in a random graph: a newcoming user joins the overlay by connecting itself to any, randomly chosen, existing node. Data that is stored in overlay nodes is queried either by flooding, or by using random walks on the graph. This results in inefficiency, since a large subset of nodes has to be queried for a content not widely distributed to be found. Despite the considerable efforts made on improving the unstructured overlays (the most recent, Gia [29]), as a response to a poor data discovery performance, the structured overlays have emerged.

Structured overlays are built in a controlled fashion. In these systems, to each data item and to each member a unique logical identifier is assigned. Based on the member's identifier, new members join by attaching to a well defined existing member, which results in a highly structured graph. But not only is the placement of new members precisely determined, also is the placement of data items. These systems rely on distributed hash tables (DHTs) to map the key (the identifier of data items) to the node in charge of storing that particular key and that particular data item. This enables efficient search for exact queries, since a data item, given its key, can be found in only $O(logN)$ hops. In addition, the total of only $O(logN)$ neighbors should be maintained per each node.

Some recent studies [53] have shown that the churn (the rate at which nodes join and leave the network) as well as the heterogeneity of users are high. Many argue that unstructured overlays can handle such an environment more efficiently. In addition, they provide more efficient search for complex queries of popular data items than their structured counterparts. However, in [22] Castro *et al.* compared structured and unstructured graphs via detailed simulations. A hybrid system has been designed that constructs a structured graph, however data placement and search mechanisms are the same as those deployed in unstructured overlays. Simulation results based on a real-world samples indicated that the hybrid system can support complex queries with lower message overhead while providing higher query success rates and lower response times than systems based on unstructured graphs.

These structured overlays can be used for AL multicast to be built on top of them as an application. Many such proposals have emerged. AL multicast can be built on top of the structured overlays as an application. What all the proposals of this group have in common is that to each member a logical address from some abstract coordinate space is assigned. As a consequence, the data distribution trees are embedded in the overlay, and no routing protocol is needed. This enables these schemes to scale to very large group sizes (thousands of members), without being restricted to single-source applications, which is the major advantage of these schemes.

The main drawback is their degraded performance compared to that of the protocols of the first two groups. Their worse performance is a consequence of the non-optimal match between the overlay and the underlying substrate. Namely, the end users do not possess topology information generally available to network routers. Therefore, when connecting to each other into an overlay, often they do not take the underlying topology into account. In that case the two neighboring nodes in the overlay may be separated by many hops on the IP layer. Consequently, even if optimal in the number of application layer hops, a path between the source and the destination on the overlay may consist of a large number of IP-layer hops.

Generally, there are three possible approaches toward integrating the topology awareness into the overlay: Proximity routing, Topology-based nodeId assignment and Proximity neighbor selection. These techniques will be briefly explained further in this chapter. However, achieving the congruency of the overlay with the underlying IP-layer network in the optimal way remains a difficult problem, and the integration of topology information into the overlay remains partial and incomplete.

6.3.1 CAN-based Multicast (MCAN)

CAN overlay network

As the name reveals, this multicast scheme [86] exploits the underlying structured overlay network CAN [85]. The nodes in a CAN are identified in a d-dimensional Cartesian coordinate space on a $(d+1)$-torus. Every node in a CAN holds an individual zone (see Figure 6.6), determined by its coordinates. The routing table in each node consists of IP address and the zone coordinates of that node's neighbor, where two nodes are considered neighbors if their zones overlap in all but one dimension. This implies that each node maintains the routing table with $O(d)$ entries, resulting in the routing table independent of the network size. The routing of the message is performed in a greedy forwarding fashion: the packet is sent to the neighbor with coordinates closest to those of the destination.

The overlay is formed in the following way: a joining node first discovers a node already participating in CAN by using the bootstrap mechanism. Next, it chooses randomly a point with (x, y) coordinates in the CAN space. Via the bootstrap node, it sends a join request message to a node occupying the zone with (x, y) coordinates. The zone in which the point (x, y) lies is then split in two: one half is kept by the current holder of that zone, and the other half is allocated to the new node[2]. Subsequently, both nodes update their routing table and this change is announced to their neighbors, to update their routing tables as well. When a node leaves, it hands the zone over to one of its neighbors.

Topology Awareness in CAN overlay network

The CAN overlay allocates nodes to zones at random, i.e. there is no relation between node coordinates and the underlying topology. Hence, neighbors on a CAN are not necessarily physically near each other in the Internet. Consequently, routing may be inefficient. The topology awareness technique that has been implemented in CAN is coined topology-based nodeID assignment. In this approach, topology information

---

[2] In a two-dimensional CAN, the zone is split first along x, and then along the y axe.

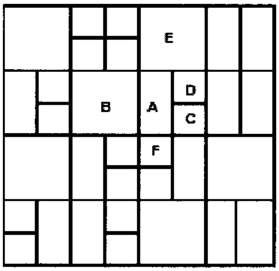


Figure 6.6: A two-dimensional CAN overlay network example.

is integrated in the overlay construction: the nodes that are close to each other in the underlying network are neighboring each other in overlay. The designers of CAN proposed in [87] landmark-based placement for accounting for the underlying topology when creating CAN. In this mechanism each joining node probes a set of landmark machines, estimates network distances by measuring the round-trip-time and orders the rtts according to increasing order. The CAN space is then divided into evenly sized bins and, rather than choosing a CAN node among all nodes randomly, the split node is randomly chosen within the bin area. These bins are clusters of nodes with same landmark ordering. Hence, the nodes topologically close to each other are likely to have the same ordering and will be located in the same part of CAN space.

The Forwarding Algorithm in CAN-based Multicast (MCAN)

The CAN overlay can easily be exploited for multicasting. If we assume that CAN nodes, or their subset, form the multicast group, then the simplest way to achieve multicasting would be to simply flood the message to all the participating nodes. Since the simple flooding algorithm induces a large number of duplicate packets, a more efficient forwarding algorithm has been proposed in [86]. Nevertheless, even the improved algorithm generates a substantial number of duplicate packets. This algorithm we describe below and further refer to as MCAN1.

Forwarding algorithm (MCAN1):

Origin forwarding rule: The source that generates a new message forwards that message to all its neighbors (Figure 6.7(a)).

General forwarding rule: If a node receives the message *along* the y-axis, then it will further forward the message along the y-axis in the direction going away from the source, and to neighbors in both directions along the x-axis (In Figure 6.7(b), node H forwards the message to I, J,K, and G). Otherwise, if a node receives the message *along* the x-axis, it will forward it only to neighbors along the x-axis, in the direction away from the source (node C in Figure 6.7(b)).

The halfway rule: A node does not forward a message along a particular direction if the message has already propagated halfway across the space from the source coordinates along that dimension. The halfway rule is illustrated in Figure 6.7(c) (G and H will not sent to K, A will not send to N, etc). This rule prevents the flooding from looping round the back of the space.

Cache suppress rule: A node caches the sequence numbers of received messages and does not forward the same message more than once.

The corner filter rule: This rule is built upon the fact that every node knows the zone coordinates for each of its neighbors. A node only forwards the message to its neighbor if its zone is in contact with the *corner* of that neighbor. The corner of a node is determined in the following way: assume node X receives the message along a particular axis. Assume further that it borders node Y along the orthogonal axis (in the opposite direction from which it received the message). The corner C_Y of Y is then the lowest coordinate of Y in that (orthogonal) dimension. In Figure 6.7(d), C_M is the corner of node M, while C_F is the corner of node F. Thus, nodes M and F receive the message only from nodes J and D respectively, since J and D's zones are in contact with M and F's corners.

6.3.2 SCRIBE

Here we will briefly describe the Scribe multicast algorithm together with its Pastry overlay network. Only routing mechanisms, relevant to our analysis are explained. For more details, we refer to [40].

Scribe is a tree-based application layer multicast mechanism built on top of Pastry. Scribe builds a single multicast tree for the whole group. Each group has a unique *groupId* and a root of the group tree. The tree is created by combining Pastry paths from each group member to the tree root. A node wishing to send a message to the group delivers the message first to the root. From the root, multicast messages are then distributed along the multicast tree.

6.3.3 Pastry

In the Pastry overlay network, each node has a unique identifier (*nodeId*). The identifiers are chosen out of the 128-bit circular space and assigned to the nodes. *NodeIds* are presented as a sequence of digits in base 2^b (b is a configuration parameter with typical value 2 or 4).

To route a message, a node uses a leaf set and a routing table. A leaf set consists of typically $2b$ or $2 \times 2b$ entries, which are filled with *nodeIds* numerically closest to the current node's *nodeId.* A routing table consists of $128/b$ rows, with $2b$ columns per each

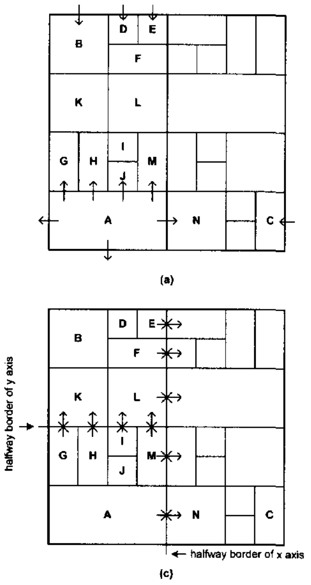


Figure 6.7: The original multicast forwarding algorithm in CAN.

row. The routing table is constructed in the following way: the entry in the n-th row and k-th column of a node's routing table contains a *nodeId* that matches the current node's *nodeId* on the first n digits, and its $(n+1)$-th digit is equal to k. If there is no *nodeId* that satisfies this criterion for an entry, the entry is left empty. An example of a routing table for a node with *nodeId* 12030321 is illustrated in Figure 6.8.

The routing in Pastry is performed as follows: upon receiving a message, a node first searches for the destination key in its leaf set. If the leaf set does not contain that key, the node starts searching in its routing table for the node whose *nodeId* matches the destination key in at least one digit more than the current node does. If such a node is not found in the routing table, the current node will look back into its leaf set and forward the message to a node whose *nodeId* matches the destination key in the same number of digits as the current node, but is numerically closer to the key.

If the number of Pastry users is sufficiently large, there might exist several nodes that meet the criterion for an entry. The node will then be chosen randomly out of all candidates satisfying the criterion. Alternatively, a proximity neighbor selection technique can be applied: an entry in the routing table includes the *nodeId* of the node topologically nearest to the current node, of all possible candidates. In the original Pastry design, Rowstron *et al.* [89] assumed that a function exists that enables each Pastry node to determine its relative "distance" to another node, given the node's IP address.

6.3.4 Bayeux

Bayeaux [112] is very similar to Scribe, but uses Tapastry [111] as an underlying overlay. They also differ in approach to supporting replication. Tapestry uses a distributed algorithm called surrogate routing for mapping keys uniqely to an existing node in the network, in case a node's routing table has no entry for a node matching the n-th digit. A minor difference lies in the direction in which digits are resolved (from right to left).

6.3.5 Delaunay triangulation

This protocol builds overlay topology which satisfies Delaunay Triangulation property [71]. It targets very large multicast group sizes.

How it works: Each member is assigned logical (x, y) coordinates that can reflect geographical location of the nodes. Based on these coordinates, the nodes are connected so that topology created consist of triangles that satisfy Delaunay Triangulation property. A Delaunay Triangulation (DT) for a set of nodes is such a triangulation, where for each circumscribing circle of a triangle formed by three nodes, in the interior of the circle there is no other node (see Figure 6.10). Delaunay Triangulations can be easily determined locally, since it has been proven in [91] that DT satisfies the locally equiangular property: a triangulation is a DT if the minimum internal angle of

NodeId 12030321

Leafset	Smaller	Larger
	12030221	12031210
	12030000	12031001
	12030123	12031310
	12030220	12032001

Routing Table

	0	1	2	3
0	01230100	1	23012101	32132012
1	10233221	11000123	2	13201323
2	0	12103320	12231022	12332101
3	12003211	12012301	12023002	3
4	0	12031310	12032211	12033001
5	12030001	12030121	12030221	3
6			2	
7		1		

Figure 6.8: Leaf set and Routing table for node in Pastry with *nodeId* 12030321. Depending on the destination key, the node chooses the next hop from the routing table according to the longest prefix rule. E.g., for a destination key 31032030, 12030111, 12030000 the next hop would be 32132012 (row 0 column 3 in routing table), 12030121 (row 5 column 1), and 12030000 (in leaf set) respectivelly.

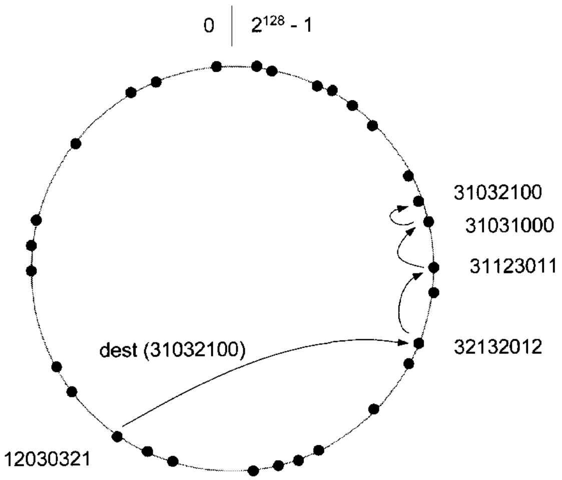


Figure 6.9: Routing in Pastry.

the adjacent triangles is maximized. The equiangular property is illustrated in Figure 6.11. Triangles ΔABC and ΔABD, which share a common link AB, satisfy the locally equiangular property, since their minimum internal angle is not smaller than the minimum internal angle of triangles ΔACD and ΔBCD. The equiangular property ensures that nodes geographically close to each other will be connected in the topology. By enforcing this property for each node, the topology can be constructed and maintained in a distributed fashion. Another attractive property of DT is that there is a set of non-overlapping routes between any pair of nodes, which enables a good scalability of this protocol. Finally, no tree building protocol is needed. Once the overlays is created, as we will demonstrate below, the routing information is embedded in it.

The overlays are created as follows: the joining node is bootstrapped to any node already in the DT overlay. This node subsequently performs a neighbor test on the joining node: it verifies whether an newcoming member can become a neighbor or not. Based on the coordinates of the current neighbors and joining node, a testing node verifies whether a link toward a joining node would satisfy the equiangular property with the current neighbors. If so, a new node is accepted as a neighbor of the testing node. Otherwise, the testing node redirects the newcomer to the node whose coordinates are closer to the coordinates of the newcomer. The process ends once a new node passes a neighbor's test. Hence the only state information to be maintained by each node is confined to that of its neighbors.

As mentioned above, there is no tree building protocol required in this approach.

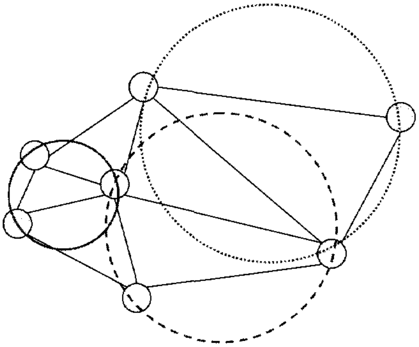


Figure 6.10: Delaunay Triangulation.

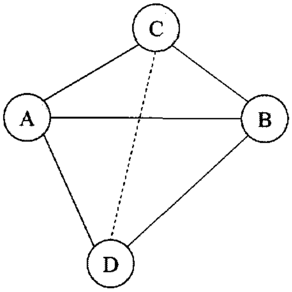


Figure 6.11: Locally equiangular property.

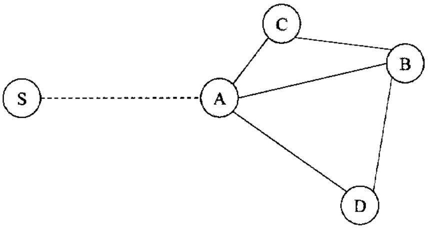


Figure 6.12: Compass routing.

i	0	1	2	3	4	5	6	7
$Bin(i)$	000	001	010	011	100	101	110	111
$G(i)$	000	001	011	010	110	111	101	100

Table 6.1: Binary and Gray code comparison.

For each sender the next hop is determined locally, as each node is able to locally determine its children nodes with respect to the tree rooted at the sender. Each node accomplishes this by implying the "compass routing" procedure on its coordinates and the coordinates of its neighbors and the sender. Compass routing is demonstrated in Figure 6.12. Node A determines that its neighbor B will be its child node in a tree routed at S, since selecting A leads to smaller angle from B to S than selecting C or D. While compass routing in planar graphs may result in routing loops, this is not the case for Delaunay triangulations [66]. Hence, no loop detection mechanism is required.

6.3.6 HyperCast

How it works: HyperCast protocol [70] organizes multicast members in the logical n-dimensional hypercube structure, a structure containing 2^n nodes (Figure 6.13). To each node a label in the form of a binary string is assigned, (e.g., "010"), indicating the position of the node in the logical hypercube. The bit strings are assigned in such a way that two nodes will be neighbors in the overlay, if their bit sequences differ in exactly one bit (Gray code). Each node then needs to maintain the table with the addresses of its (maximum n) neighbors. The relation between the binary and Grey code is shown in Table 6.1.

Similar as in DT protocol, HyperCast requires no multicast routing protocol. Arranging the nodes according to the Gray code ensures that the routing tree can be embedded even in the incomplete hypercube (when there are less members than 2^n, which is most often the case) with the use of a very simple algorithm. This algorithm

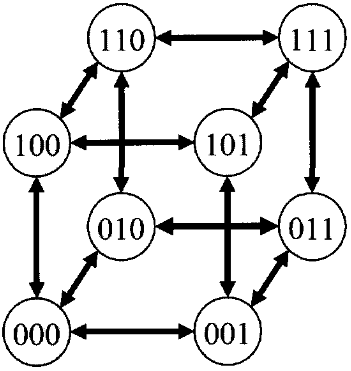


Figure 6.13: Hypercube.

enables a node with a label $B(i)$ to compute the label of its parent node in the tree with a root label $B(r)$ by using only the labels $B(i)$ and $B(r)$ as input, and to do that by inverting only a single bit. This algorithm works as follows: Let us assume that $B^{-1}()$ is the inverse function of function $B()$ that assigns a number to a binary label, i.e., $B^{-1}(B(k)) = k$. Let us further denote with $B(i) = I = I_n \dots I_2 I_1$ and $B(r) = R = R_n \dots R_2 R_1$. the label of the i-th and r-th node in the Gray encoding, respectively. The label of the parent node of node I in the tree rooted at R is computed in the following way: *if* $B^{-1}(I) < B^{-1}(R)$, *the least significant bit in which* I *and* R differ should be inverted. Otherwise, the most significant bit in which I and R differ will be inverted. An example of the routing tree in an incomplete hypercube with 7 nodes, for root node 000 is depicted in Figure 6.14. Forwarding of packets is straightforward: a message is sent to a neighbor that has one bit more of overlap with the destination label. For details on the join procedure and node failure recovery we refer to [70].

6.4 Hierarchical multicast

Algorithms of this group arrange multicast members into hierarchical structures, which is an imperative for obtaining better scaling characteristics. In Kudos [61], a two layer hierarchy is constructed, with a mesh-based sort of protocol on each of the layers. NICE [11] sets up a multi-layer hierarchy, where on each layer hosts are distributed into a set of clusters.

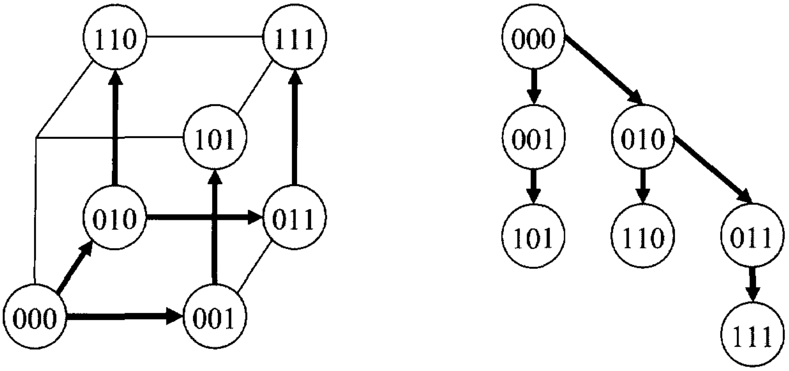


Figure 6.14: Tree with 000 as root embedded in an incomplete Hypercube.

6.4.1 Kudos

Kudos [61] protocol introduces a two-layer hierarchy to increase scalability, with a mesh-based mechanism at each layer. Hence, it also can be classified as a mesh-based protocol.

How it works: Kudos organizes members into clusters (sets of nodes). If there are totally N nodes that form the overlay, Kudos will arrange them in approximately $N^{1/2}$ clusters each containing approximately $N^{1/2}$ nodes. In each cluster the node that is closest to all the other nodes in that cluster is determined and is referred to as the cluster head. All the other nodes in the cluster are referred to as children. In each cluster the nodes belonging to that cluster form an overlay network, using any mesh-based protocol (e.g. Narada a[31]), independently of the nodes in other clusters. Top layer in this two layer hierarchy is formed by cluster heads from the layer below (see Figure 6.15). The cluster heads in the top layer will also connect themselves into a network in the top layer, again with the use of some mesh-based algorithm.

There are three clustering operations that this protocol performs: migration, splitting and diffusion. Migration is related to the join process. When a new node wants to join the network, it is bootstrapped to a randomly chosen cluster. From the head of that cluster the newcomer receives the list of the other head nodes. Among them it selects several and evaluates the latency toward them. If a child detects a head that is considerably closer to it than the current head, it will migrate to that cluster. In order to limit the number of probes the child node executes, the child will only probe those heads for which the latency toward the current child's head is less than twice the latency between the child and it's current head.

If a cluster outgrows the size of twice $N^{1/2}$, the cluster will be split in two. The

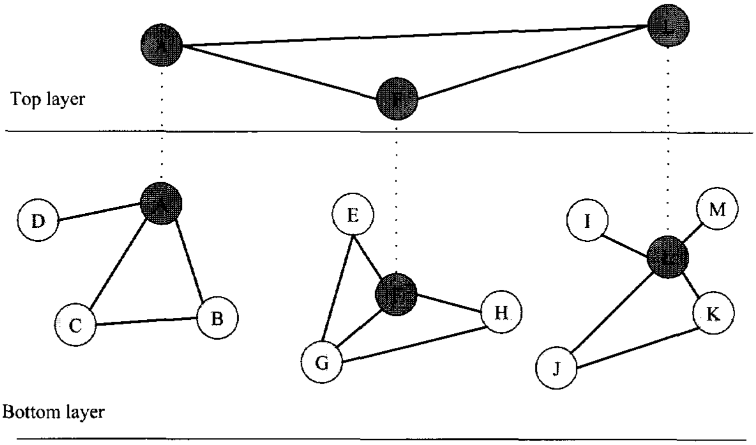


Figure 6.15: Two level hierarchy in Kudos.

head of the old cluster remains the same. The old head sends the latency information to all its child nodes. Every child responds with the computed number of other child nodes that are closer to it than the the old head is. Based on these responses, the old head appoints the head for the new cluster. Each child node in the former cluster based on the information it receives.

Whenever the size of the cluster diminishes to less than the half of $N^{1/2}$, due to node failure, member leave or migration, the cluster diffuses. The remaining nodes migrate to the neighboring clusters.

Due to the clustering and hierarchical organization this protocol achieves higher scalability and lower management overhead than its one layer equivalent, since measure probes are run across smaller groups and member failures effect smaller groups. The price paid for this is the efficiency, since the children nodes belonging to different clusters cannot form overlay links among each other.

6.4.2 NICE

The goal of the NICE protocol [11] is to organize group members into a hierarchical, layered structure.

How it works: All group members are assigned to the lowest layer L_0, and they are grouped into clusters. The nodes within a cluster periodically exchange refresh messages containing the information on latency among them. The arranging of nodes into the clusters is performed according to this latency. The clusters are of size between k and $3k-1$ nodes, where k is a constant. The node in each cluster that is closest to all

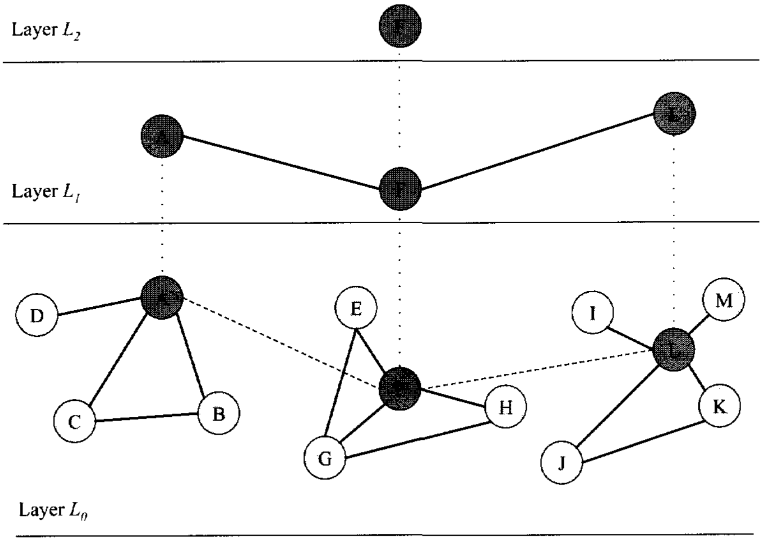


Figure 6.16: NICE hierarchical structure and clustering.

the other nodes in that cluster is designated as the cluster leader. The cluster leaders of all clusters at the layer L_i form clusters in the layer L_{i+1}. This process continues until the layer is reached with only one member, resulting in maximum $log_k N$ layers (see Figure 6.16).

The joining procedure works in the following way: When a new node wants to join, it contacts the *rendezvous point*, from which it receives the information on the members in the highest layer (L_n). Among those members, it finds the member closest to itself in terms of latency. This member, the cluster leader of a cluster in layer L_{n-1}, notifies the newcomer on the members in its cluster on layer L_{n-1}. The newcomer searches in that cluster and finds the closest member to itself again. This procedure is repeated until the newcomer finds the adequate cluster in the layer L_0.

If upon joining of the new host that cluster exceeds the size of $3k-1$, it will be split into two clusters of minimum size $3k/2$, in such a way that the maximum of the radii among the two clusters is minimized. If a host wants to leave, it notifies the members of all the clusters it belongs to.

Also in NICE no routing protocol is needed, since the source-specific trees are embedded into the topology itself. An example of such a tree in a NICE structure with a cluster size $k = 4$ is given in Figure 6.17. A source sends a data packet to all its peers in the overlay topology (node G belongs only to layer L_0, so it only sends the packet to

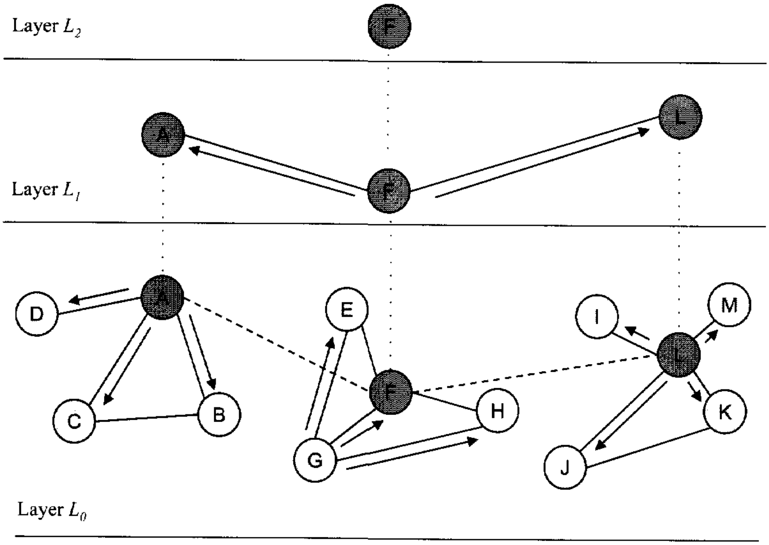


Figure 6.17: Multicast routing in NICE.

nodes E, F and H). Assume that the sending node and the receiving node belong to the same cluster at the layer L_i. If and only if the receiving member is the cluster leader of the cluster C_i (where C_i is its cluster on the layer L_i) it will distribute the packet to all the other members of the cluster C_k, $k \neq i$. In the example in Figure 6.17, node F, the cluster leader of a cluster on the layer L_0, receivers a packet from node G, one of its peers in that cluster. Therefore, it distributes the message to the other members of its cluster in layer L1, nodes A and L. Nodes A and L will subsequently disseminate the message to all the other nodes in their clusters on the levels below.

6.5 Conclusion

This chapter has presented seventeen different mechanisms for AL multicast realization. All protocols have been classified in four categories based on the type of topology created for communication between end hosts.

6.5.1 Efficiency

In general, it is difficult to analytically compute the efficiency for most protocols as most of the results are obtained either via simulation or empirically. The performance also varies according to group sizes and the characteristics of the underlying topology. In most cases different algorithms are not compared under the equal conditions. This has been a trigger for our analysis, presented and discussed in the following chapter.

6.5.2 Scalability

The scalability of these protocols is determined by control overhead. Control overhead refers to all the non-data traffic that traverses the network. It includes traffic required to maintain the overlay as all application layer multicast solutions require nodes to exchange refresh messages with their neighbors and the probe traffic and other active measurements performed during the overlay self-organization and maintenance process. The average overhead per node is often used as an indicator of the scalability of the protocol. Another metric is the amount of information kept by each node on the overlay. State information maintained in each node has to be periodically updated to reflect dynamically changing network environment. This implies that a node maintaining more state information will generate more update messages, contributing to additional protocol overhead. This metric therefore directly impacts the scalability of a technique. Information maintained by each node includes routing tables and group state. Protocols which require the full set of member state to be stored in each node are thus not as scalable as those which only maintain partial group member state.

An intuitive, but simple approach, which can provide a quick insight into the scalability of each technique, is the number of members each of the proposed protocols can support. The protocols we described in this chapter can be classified in four groups based on their scalability:

Low scalability (several tens of multicast members): ALMI.

Medium scalability (several hundreds of multicast members): Narada, BTP, Yoid, HMTP, TBCP.

Large scalability (several thousands of multicast members): CAN, Scribe, Scattercast, Overcast, Bayeaux, Kudos.

Very large scalability (several tens of thousands of multicast members): NICE, DT.

Chapter 7

Hopcount in Application Layer Multicast

7.1 Problem statement

In Chapter 6, we have classified and reviewed several AL multicast schemes. AL multicast enables rapid and seamless deployment of multicast applications. However, the price that it has to pay is the performance penalty, because a packet may be replicated and forwarded on the same link more than once. The objective of this chapter is to determine and compare the efficiency of different application layer multicast schemes under the same conditions. We focus on several schemes that belong to two latter classes presented in Chapter 6, due to their superior scalability.

In the early approaches, overlay networks have been created without considering the underlying Internet topology. More recently, methods have been developed for integrating the information of the underlying network into the overlay. Currently, these methods can be classified into three groups: proximity routing, topology-based nodeId assignment, and the proximity neighbor selection. Section 6.3.1 and Section 6.3.2 discuss several such techniques. Some experiments have suggested that the proximity-neighbor selection attains the optimal results [23].

The goal of this chapter is twofold:

1. We evaluate the performance of three scalable application layer multicast algorithms, CAN-based multicast (MCAN) [86], Scribe [40] and NICE [11]. As a performance metric the number of hops has been used. We compare these schemes mutually, as well as to multiple unicast connections (we denote them further with m-unicast) and IP multicast. Furthermore, we introduce and evaluate modifications to MCAN that lead to a better performance in terms of the number of hops (duplicate packets). For clarity, with MCAN1 we denote the original, and with MCAN2 our modified algorithm. We perform this evaluation both via extensive simulations, as well as via experiments

on PlanetLab [4].

2. We investigate the influence of the underlying topology awareness on the performance of application layer multicast. In our study, for each of the implemented algorithms, we have considered two extreme situations: no topology awareness, and absolute topology awareness. Whereas under the no topology awareness conditions nodes in overlay are placed randomly, without taking the underlying IP-layer topology into account, the term absolute topology awareness indicates the ideal situation in which the complete knowledge of the underlying substrate is attained and used in the creation of the overlay.

To the best of our knowledge, the only study similar to ours has been provided by Castro *et al.* [24]. However, our study differs from [24] in several aspects. First, our goal is not to estimate the underlying topology. Instead, we assume that the underlying topology is either completely known to the joining overlay nodes (absolute awareness) or completely unknown. In this way, we try to establish the upper and lower boundaries for the hopcount of these three algorithms. Secondly, we use the hopcount as a metric, since it is an important quantity from a network point of view. Third, we perform our simulations on a very large number of different substrates (up to 10^5) which ensures the statistical credibility of our results. Fourth, we introduce the modifications to MCAN algorithm that lead to improved results for hopcount compared to the original algorithm. Finally, we evaluate our results via experiments on the PlanetLab network.

This chapter is organized as follows: Section 7.2 introduces our modifications to the CAN-based multicasting algorithm. In Section 7.3 we first explain the simulation designs of six compared schemes: MCAN with and without improvements, Scribe, NICE, m-unicast and IP multicast. The same section further presents and discusses the simulation results for different types of the underlying topology. In Section 7.4 the results of the measurements on PlanetLab are presented. Finally, we conclude in Section 7.5.

7.2 Forwarding in CAN based Multicast (MCAN)

The forwarding algorithm used to realize multicasting in CAN and proposed in [86] has been described in Section 6.3.1. Nevertheless, even the improved algorithm generates a substantial number of duplicate packets. In order to further reduce the number of duplicate packets we introduce certain modifications to this algorithm. In this Section we describe our modifications. The modified algorithm is explained on an example of a two-dimensional CAN, and illustrate it in Figure 7.1.

Multicast CAN Modified Forwarding Algorithm (MCAN2)

Origin forwarding rule: The source that generates a message forwards the message to all its neighbors along the x-axis, *but only one neighbor per direction along the y-axis.* In Figure 7.1(a), A forwards the message to C and N along the x-axis, but only to H and only to E in positive and negative direction of y-axis respectively.

General forwarding rule: A node that receives the message along the y-axis, forwards the message to all the neighbors along the x-axis. However, it will forward the message to only one neighbor in a particular direction along the y-axis (the direction away from the source). If there are several neighbors in that direction, the neighbor to which the message will be sent is chosen randomly. A node that receives the message along the x-axis, forwards it to all the neighbors along the x-axis that lie in the direction *opposite* of that from which the node received the message (e.g. node C and node D in Figure 7.1(b)).

The halfway rule and the cache suppress rule: These rules remain unmodified.

The corner filter rule: In the modified algorithm, the corner filter rule is applied only on forwarding along the x-axis. In Figure 7.1(d) M only receives the message from J since M's corner is in contact with J's zone. However, in the y-axis forwarding, even though E's zone does not touch F's corner, E will forward the message to F.

7.3 Evaluation via Simulations

In our simulations, we confine ourselves to random graphs of class $G_p(N)$ [18]. We first generate a graph consisting of $N \geqslant 100$ nodes, representing the routers. For each graph of N nodes, we define the number of multicast users m in the network, such that a ratio $\rho = m/N$ takes the value from the set $\rho = \{0.05, 0.1, 0.2, 0.5, 0.7, 0.9\}$. For each N, 10^5 different topologies are generated. In the simulations of Scribe, before defining m, we first define the number of nodes N_{Pastry} that constitute the Pastry network. The multicast members in Scribe form a subset of nodes participating in Pastry, such that the $N_{Pastry} = m,\ 2*m\ ,4*m$.

Two different scenarios for the members location have been considered: in scenario a, some multicast members (or Pastry nodes) may belong to the same router, while in scenario b, each member is attached to a different router.

In each underlying topology, the following nine different multicast schemes have been implemented: multiple-connections unicast, IP multicast, the original MCAN with and without the topology awareness (MCAN1 and MCAN1_top), the modified MCAN with and without topology awareness (MCAN2 and MCAN2_top), Scribe without and with the absolute topology awareness (Scribe and Scribe_top) and NICE without topology awareness (NICE). For each mechanism and each underlying topology, the number of hops in the path is computed and stored in a histogram. In a multiple-connections

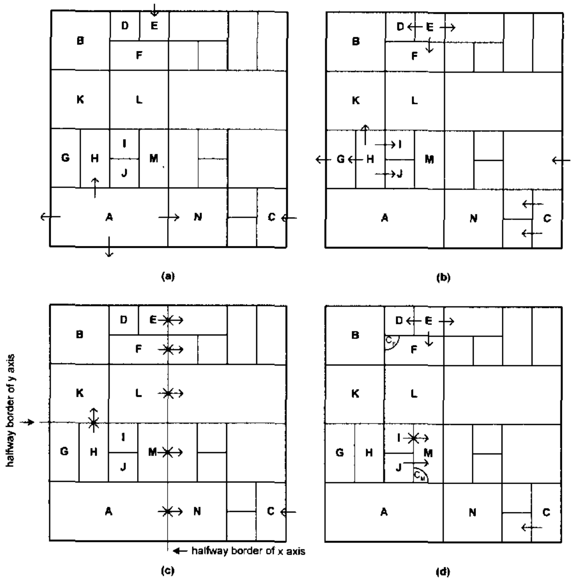


Figure 7.1: The modified multicast forwarding algorithm in CAN.

unicast, the total hopcount is the sum of the hopcounts along shortest paths from a source to each of the $m-1$ destinations individually. For IP multicast, we assumed that the message is disseminated along the Shortest Path Tree[1], since the most of the current IP multicast routing protocols forward packets based on the (Reverse) Shortest Path. The total hopcount is equal to the sum of the links in a multicast tree. For MCAN, in addition to the total number of hops traversed, an effective number of hops has been computed and stored. An effective number of hops is computed by only including the hops traversed in order for all destinations to be reached, without including the forwarding paths of the duplicate messages.

7.3.1 MCAN on a random graph

We have assumed for simplicity that all nodes in a CAN form a multicast group, and we created a two-dimensional CAN. In order to evaluate MCAN on top of a random graph, three different sets of simulations have been performed:

(*i*) Single overlay: in this set of simulations, no topology awareness is considered. For each underlying topology the CAN overlay structure is kept the same. Next, we consider two scenarios as stated above, scenario a and b. After a CAN overlay consisting of m MCAN members is generated, the m MCAN members are mapped onto each of the underlying topology in the following way: in scenario a, each MCAN member is mapped to one out of N nodes in the underlying network randomly. In this scenario, the same node can be chosen multiple times, i.e. several MCAN members can be attached to a same router. In scenario b, m out of N nodes in the underlying network are chosen randomly, and each of the m MCAN members is assigned to one of them. Hence, each member is attached to a different router. For each scenario, and each underlying topology, both MCAN1 and MCAN2 have been implemented, and their hopcounts have been stored in histograms. From each histogram, the probability density function of the hopcount was deduced, together with the mean $E[H_N]$ and the variance $var[H_N]$.

(*ii*) Multiple overlays: For each underlying topology, $r = 1000$ different samples of a CAN overlay, each consisting of m members, have been generated. For each of the overlay instances, the identical procedure as described in (i) has been applied.

(*iii*) Topology aware overlay: For each m and N, and both scenarios a and b, in each of the generated underlying topology, the identical nodes (routers) to those chosen in (i) have been selected. The CAN overlay is then constructed using the information about the distances in the underlying network. Each newcoming node learns the hopcount to all the other nodes already in CAN using the Dijkstra algorithm. The nearest node is then chosen to be the split node.

---

[1] A Shortest Path Tree is the union of the shortest paths between the source and all m destinations.

sch.	MCAN	m/ρ	1/2	sch.	MCAN	m/ρ	1/2	sch.	MCAN	m/ρ	1/2
1	single	10/0.05	1	13	mult	10/0.05	1	25	t.a.	10/0.05	1
2	single	20/0.1	1	14	mult	20/0.1	1	26	t.a.	20/0.1	1
3	single	40/0.2	1	15	mult	40/0.2	1	27	t.a.	40/0.2	1
4	single	100/0.5	1	16	mult	100/0.5	1	28	t.a.	100/0.5	1
5	single	140/0.7	1	17	mult	140/0.7	1	29	t.a.	140/0.7	1
6	single	180/0.9	1	18	mult	180/0.9	1	30	t.a.	180/0.9	1
7	single	10/0.05	2	19	mult	10/0.05	2	31	t.a.	10/0.05	2
8	single	20/0.1	2	20	mult	20/0.1	2	32	t.a.	20/0.1	2
9	single	40/0.2	2	21	mult	40/0.2	2	33	t.a.	40/0.2	2
10	single	100/0.5	2	22	mult	100/0.5	2	34	t.a.	100/0.5	2
11	single	140/0.7	2	23	mult	140/0.7	2	35	t.a.	140/0.7	2
12	single	180/0.9	2	24	mult	180/0.9	2	36	t.a.	180/0.9	2

Table 7.1: The simulated schemes 1-36. The number of node N=200. The column 1/2 stands for type of MCAN algorithm: MCAN1 is the original, MCAN2 is the modifed. Each scheme 1-36 has been simulated in both scenario a and b.

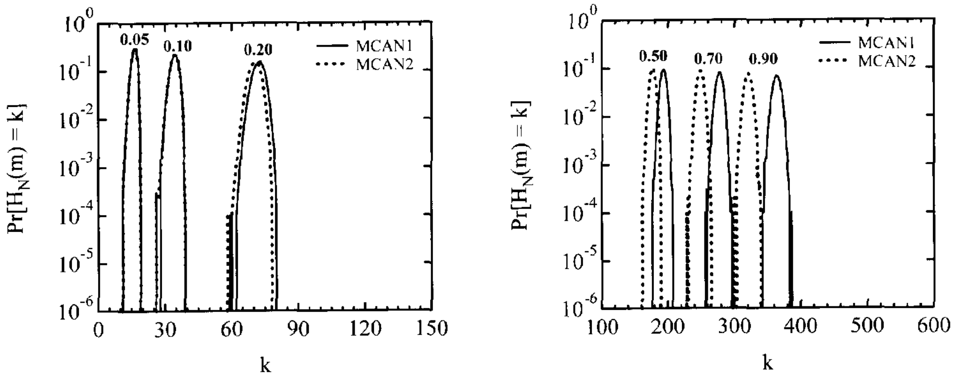
In Table 7.1 we summarize the parameters used in the simulations. In the remainder of this section we present the results of our simulations.

The pdf of hopcount in MCAN

Figure 7.2, Figure 7.3 and Figure 7.4 present the probability density functions (pdf) of the hopcount of both MCAN1 and MCAN2 algorithms in a single (schemes 1-12), multiple (schemes 13-24) and a topology aware (schemes 25-36) CAN structure respectively, for $N = 200$, and scenario a. In each Figure, (a) and (b) give the pdfs for low and high value of ratio $\rho = m/N$ respectively. Figure 7.2 suggests for for both MCAN1 and MCAN2 a similar, bell-shape form of pdf, based on simulations on a single overlay. However, the average hopcount of MCAN2 is up to 11% lower than that of MCAN1 with the increase of m.

In Figure 7.3 and Figure 7.4 we observe another interesting phenomenon: a "clustering" effect for the hopcount in MCAN1. These figures, particularly Figure 7.3, indicate that the values of hopcount of MCAN1 in multiple overlays (schemes 13-18) concentrate around several dominant values. This suggests that there seems to exist a finite number of "groups" of CAN structures.

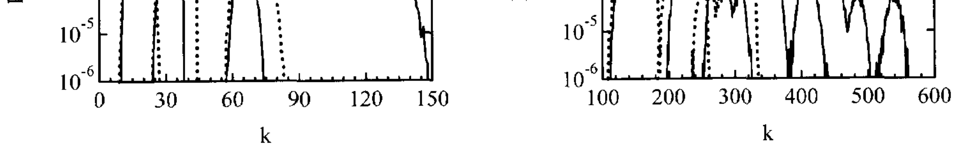
This behavior is observed in the topology-aware CAN as well (scheme 25-30), however the number of values around which hopcount concentrates is lower. The phenomenon of clustering is not reflected in the hopcount of MCAN2, neither in multiple, nor topology-aware overlays. The pdf of MCAN2 is bell-shaped, with a variance much lower than that of MCAN1. The average hopcount of MCAN1 is again 10% higher than that of MCAN2.



(a)

(b)

Figure 7.2: The pdfs of MCAN1 and MCAN2 in a single CAN overlay. (a)~schemes 1-3 and 7-9, $N = 200$, $\rho = 0.05, 0.1, 0.2$, scenario a. (b)~schemes 4-6 and 10-12, $N = 200$, $\rho = 0.5, 0.7, 0.9$, scenario a.



(a)

(b)

Figure 7.3: The pdfs of MCAN1 and MCAN2 in multiple CAN overlay. (a)~schemes 13-15 and 19-21, $N = 200$, $\rho = 0.05, 0.1, 0.2$, scenario a. (b)~schemes 16-18 and 22-24, $N = 200$, $\rho = 0.5, 0.7, 0.9$, scenario a.

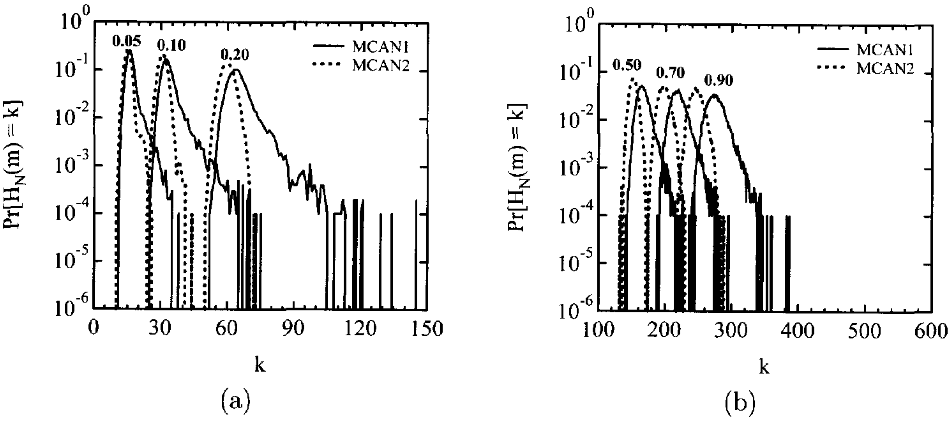
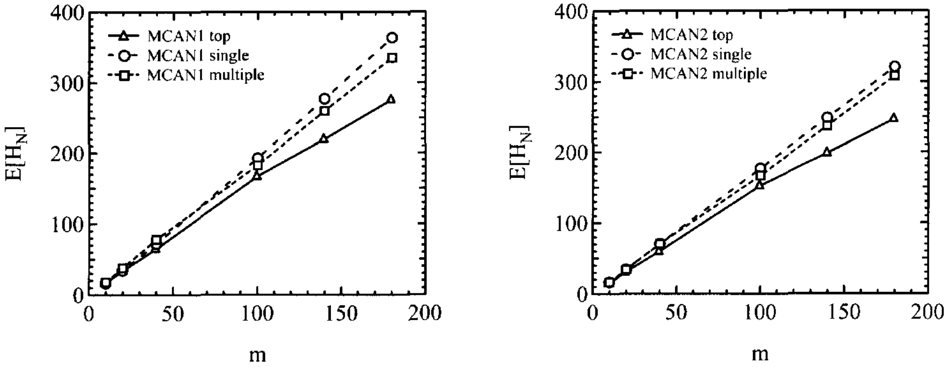


Figure 7.4: The pdfs of MCAN1 and MCAN2 in topology aware CAN overlay. (a)~schemes 25-27 and 31-33), $N = 200$, $\rho = 0.05, 0.1, 0.2$, scenario a. (b)~schemes 28-30 and 34-36, $N = 200$, $\rho = 0.5, 0.7, 0.9$, scenario a.

Effect of the topology awareness on MCAN1 and MCAN2

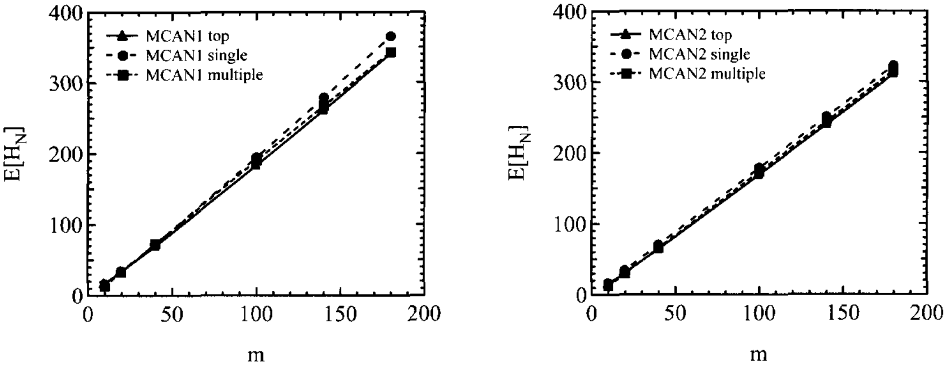
The average hopcount in MCAN1 and MCAN2 as a function of m, in a single, multiple and topology aware CAN (schemes 1-36), has been plotted in Figure 7.5 and Figure 7.6 for scenario a and b, respectively.

In scenario a, a topology aware CAN demonstrates the best performance, as expected. For high values of ρ, ($\rho \geq 0.7$) the reduction in hopcount reaches 24%. Remarkably, in scenario b, where each member is attached to a different router, the topology awareness does not seem to impact the hopcount significantly. Moreover, for a small ratio ρ ($\rho \leq 0.2$), the hopcount of MCAN1 obtained in a topology aware CAN (schemes 25-30) is higher than the hopcount in multiple overlays (schemes 13-18). A possible explanation is that when a new user joins the group, it chooses the nearest node in the CAN as the split node, which consequently may separate two nodes already neighboring each other in the CAN. This phenomenon is illustrated in Figure 7.7. We consider a small portion of the CAN overlay, where two users A and B, close to each other in the underlying network, are neighbors in the CAN space. Node C lies in another part of the underlying network. If a newcomer C is close to A on the underlying substrate, it chooses A as the nearest node already in CAN. A splits its zone and assigns half of it to C. However, this partitioning can be performed in such a way that the addition of C in the existing CAN structure will separate A and B, resulting in a higher hopcount



(a) (b)

Figure 7.5: (a) Effect of overlay creation on hopcount of MCAN1 ($N = 200$, scenario a). (b) Effect of overlay creation on hopcount of MCAN2 ($N = 200$, scenario a).



(a) (b)

Figure 7.6: (a) Effect of overlay creation on hopcount of MCAN1 ($N = 200$, scenario b). (b) Effect of overlay creation on hopcount of MCAN2 ($N = 200$, scenario b).

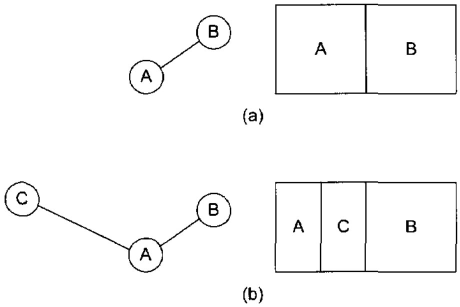


Figure 7.7: Joining of the node C increases the distance between A and B.

among them (as exemplified in Figure 7.7).

The total and effective hopcount of MCAN1 and MCAN2

In Figure 7.8 the average total and effective hopcount of MCAN1 and MCAN2 in multiple and topology aware CAN overlays (schemes 13-36), for scenario b has been plotted. The effective hops are the hops a message traverse until it reaches each of the destinations for the first time, and do not include the forwarding paths of the duplicate messages. For both multiple and topology-aware CAN (schemes 18-24, and 30-36), the ratio of the total and the effective hopcount of MCAN2 is around 1, implying that the modifications we introduce eliminate most of the duplicate messages. The effective hopcount of MCAN1 is up to 5% lower than the effective hopcount of MCAN2, suggesting, as expected, that the message forwarded with MCAN1 will reach all users in a smaller number of hops. However, MCAN2 optimizes the total number of hops traversed by the message.

7.3.2 Scribe on a random graph

For the simulations of Scribe, exactly the same random graphs as those used in the analysis of MCAN in the previous subsection have been generated. In order to simulate Scribe, we first need to define a Pastry network. A number of Pastry nodes N_{Pastry} is determined, and random *nodeIds* in the range $[0, 2^{128} - 1]$ are assigned to those nodes. Since we only simulated a small number of users ($N_{Pastry} \ll 2^{128}$), we confined ourselves to $b = 2$. The leaf set of each Pastry node consists of 8 entries. Again two different scenarios have been simulated: scenario a, in which several Pastry nodes may belong to the same router, and scenario b, where each Pastry node is attached to a different router.

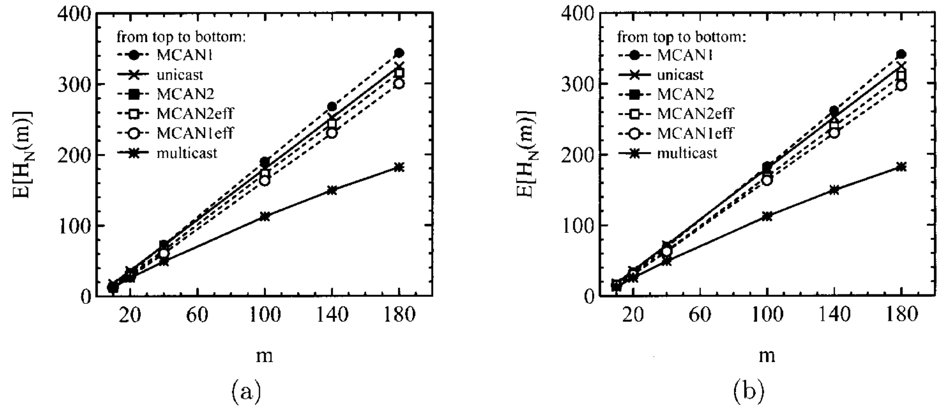


Figure 7.8: (a) Comparison of total and effective hopcount in MCAN1 and MCAN2 for multiple CAN overlays (scheme 13-24). (b) Comparison of total and effective hopcount of MCAN1 and MCAN2 in topology aware CAN overlays (scheme 25-36).

After the Pastry network has been defined, a subset of m Pastry nodes has been chosen, representing Scribe participants. Also, a root for the group tree has been determined. A Scribe tree is built by combining Pastry paths from each Scribe member to the root. Pastry paths are constructed based on the information on the underlying topology, as explicated in the previous chapter. Two different sets of simulations have been performed, Scribe, the Scribe algorithm without topology awareness and Scribe_top, in which the absolute topology awareness has been integrated.

Finally, the ratio $x = m/N_{Pastry}$ is varied, such that $N_{Pastry} = m$, $N_{Pastry} = 2 * m$ and $N_{Pastry} = 4 * m$.

Effect of the topology awareness on Scribe

Figure 7.9, presents the average value of hopcount for Scribe and Scribe_top in scenario a in (*a*) and scenario b in (*b*) respectively. As can be seen from Figure 7.9, the topology awareness seems to have a great influence on the hopcount in Scribe in both scenarios. or $\rho \geq 0.1$ the average hopcount is up to 40% lower than that of Scribe_top. For smaller values of ρ the difference in average hopcount in Scribe and Scribe_top is smaller. This figure further indicates that, while in the performance of fully topology aware Scribe is comparable to IP multicast in scenario b, this is not the case in scenario a. Furthermore, in scenario a, the hopcount of topology unaware Scribe is higher than that of unicast.

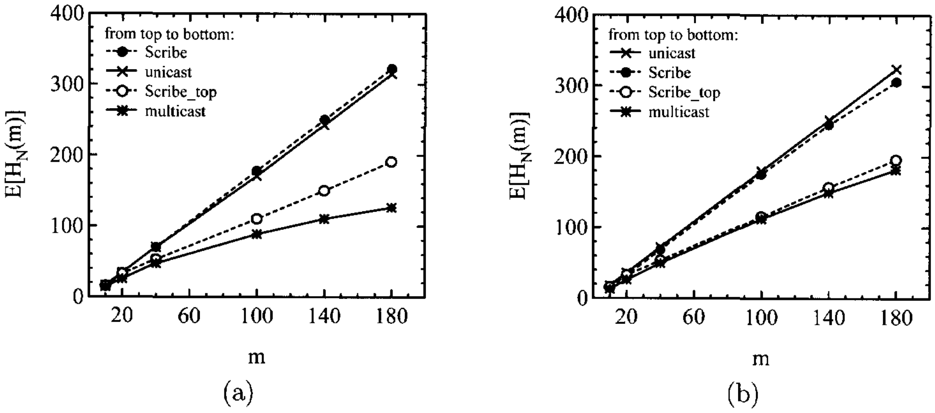


Figure 7.9: (a) Average hopcount of Scribe and Scribe_top ($N = 200$, $N_{Pastry} = M$), scenario a. (b) Average hopcount of Scribe and Scribe_top ($N = 200$, $N_{Pastry} = M$), scenario b.

Influence of the ratio $x = m/N_{Pastry}$

Figure 7.10 illustrates Scribe performance under the variation of the number of Pastry nodes (N_{Pastry}), where $N_{Pastry} = m$, $2{*}m$, $4{*}m$, and the number of users $m = 10, 20, 40$. As Figure 7.10(a) displays, the performance of Scribe is affected by the variation of these parameters. If only a half of all Pastry nodes participate in Scribe application, the difference in hopcount reaches 16%. For a number of Pastry nodes four times higher than the number of Scribe participants, the hopcount is increased up to 28%. For Scribe_top, the influence of N_{Pastry} is negligible (Figure 7.10(b)). This is an expected result, since the routing tables in Scribe (thus without topology awareness) are filled randomly. Hence, a node will choose appropriate nodes for its entries randomly, without considering their locations. Consequently, nodes that participate in multicasting will not necessarily follow the shortest possible paths to the root. By deploying the absolute topology awareness, each Scribe_top node chooses the nearest node among many candidates. On joining a multicast group, each node will follow the shortest possible path to the root, resulting in a considerably more efficient multicast tree. Moreover, with the number of Pastry nodes increasing, the possibility of creating more efficient tree increases as well.

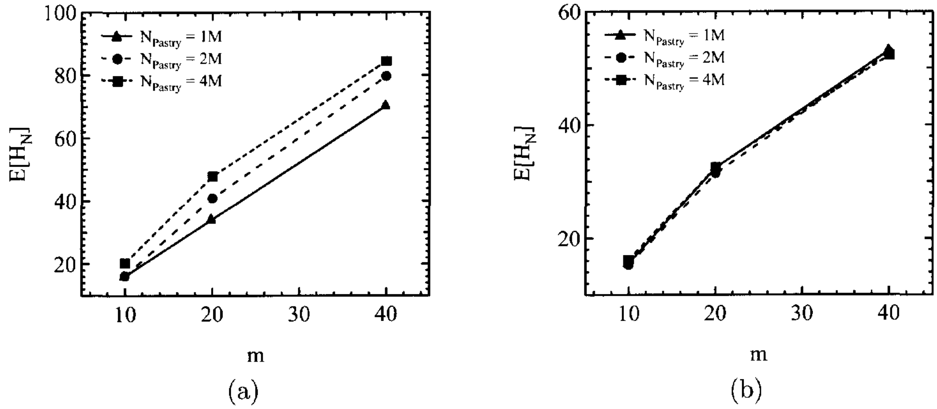


Figure 7.10: (a) Impact of N_{Pastry} ($N = 200, \rho = 0.05, 0.1, 0.2$, scenario a). (b) Impact of N_{Pastry} ($N = 200$, $\rho = 0.05, 0.1, 0.2$, scenario b).

7.3.3 NICE on a random graph

Again, under the same conditions as in simulations of CAN and Scribe, the NICE algorithm has been implemented and evaluated. For simulations of NICE, we have only simulated scenario b. In the NICE structure members are organized in layers, as discussed in Chapter 6. In each layer members are divided into clusters, with each cluster of size between l and $3l-1$ (l is a constant). The choice of cluster size bounds has also been discussed in Chapter 6. Although the original protocol assumes that members are assigned to clusters based on proximity information (to the cluster leader), in order to make our analysis complete, we first consider the case when clusters are formed in a random manner, that is, the joining nodes possess no topology awareness. We investigated the influence of the constant l on the performance of NICE.

Effect of constant l on NICE

Figure 7.11 displays hopcount in IP multicast, m-unicast and NICE, for different values of constant l. We observe that, similarly to MCAN and Scribe, the hopcount in topology unaware NICE is nearly as large as hopcount of m-unicast. This figure further suggests that the value of l does not impact the efficacy of topology-unaware NICE algorithm. A random choice of nodes that form the clusters, overlapping cluster sizes and the relatively small network size might be the reasons for not perceiving the effect

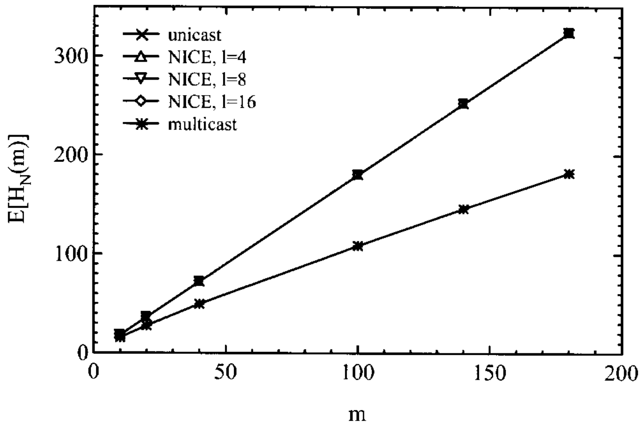


Figure 7.11: Impact of l on the hopcount in NICE ($N = 200$, scenario b).

of layering.

7.3.4 Comparative analysis on a random graph

In Figure 7.12 and Figure 7.13 the average hopcount in Scribe, Scribe_top, NICE, m-unicast and IP Multicast, together with MCAN1 and MCAN2 in multiple CANs has been plotted for both scenarios a and b, respectively. The results correspond to $N_{Pastry} = m$. As anticipated, IP Multicast achieves the best performance. Further, Scribe_top achieves the lowest number of hops among all AL multicast schemes, in scenario b even comparable to IP multicast. The hopcount of MCAN1 in both scenarios is higher than that of m-unicast. The hopcount of MCAN2 is comparable to the hopcount of m-unicast in scenario a, slightly better in scenario b. In scenario a, the hopcount of Scribe is higher than that of m-unicast. In scenario b, Scribe and NICE perform slightly worse than MCAN2 but comparable to m-unicast.

Figure 7.14 and Figure 7.15 display the average hopcount in Scribe, Scribe_top, NICE, m-unicast, IP Multicast, together with MCAN1 and MCAN2 in a topology-aware CAN (MCAN1top and MCAN2top) in both scenarios a and b, respectively. These figures seem to indicate that in scenario a the hopcount achieved with MCAN1 is lower than that of m-unicast. In both scenarios, performance of our algorithm MCAN2 is better than that of Scribe and of NICE in scenario b. In scenario-b, the results are the same as for the multiple CANs, MCAN1 obtaining the highest hopcount of all, with a slightly improved performance of MCAN1 and MCAN2.

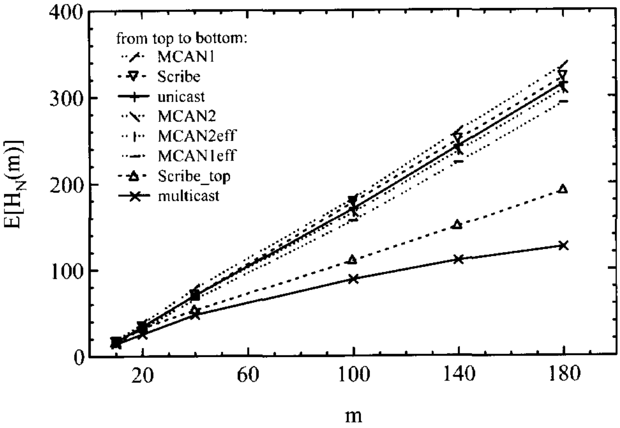


Figure 7.12: Comparison of all mechanisms in scenario a ($N = 200$) (multiple-overlay CAN).

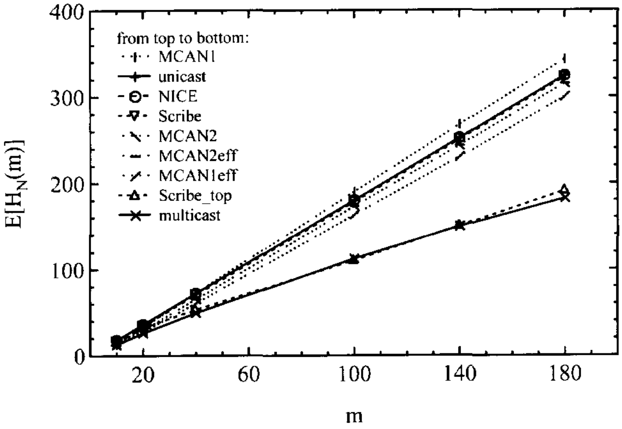


Figure 7.13: Comparison of all mechanisms in scenario b ($N = 200$) (multiple-overlay CAN).

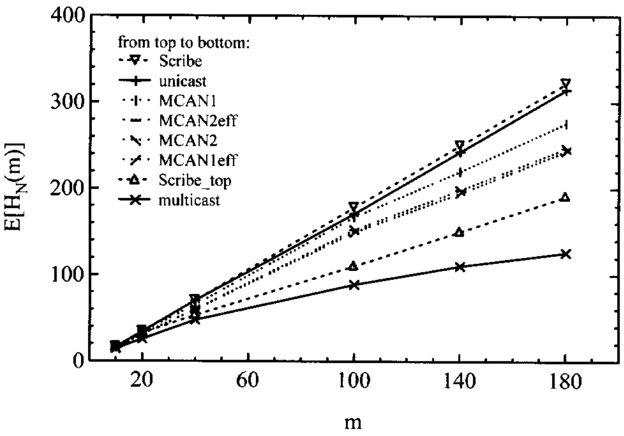


Figure 7.14: Comparison of all mechanisms in scenario a ($N = 200$) (topology-aware CAN).

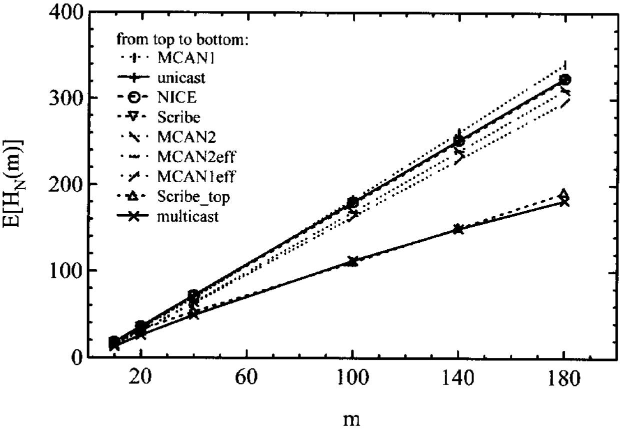


Figure 7.15: Comparison of all mechanisms in scenario b ($N = 200$) (topology aware CAN).

7.4 Evaluation via Measurements on PlanetLab

In addition to simulations, we evaluate the performance of AL multicast mechanisms under realistic conditions, by performing experiments on PlanetLab. Our experiments have been executed on November 10th 2004. At that moment, there were 445 PlanetLab nodes running on locations in USA, Asia and Europe. We have performed two sets of measurements.

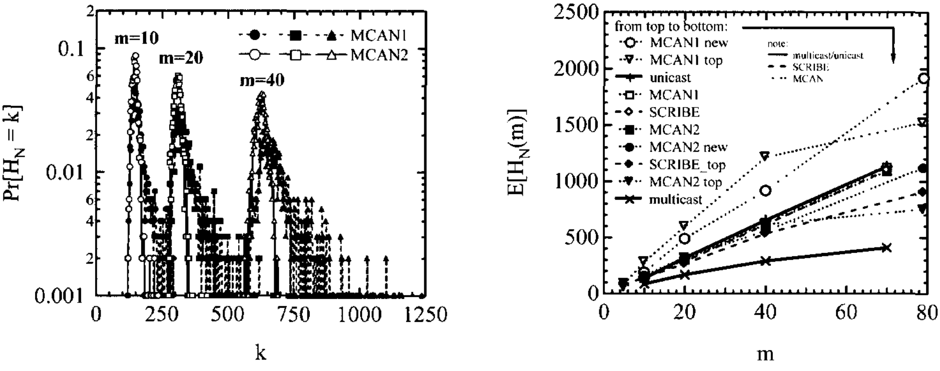
1) For the measurements, corresponding to the scenario b, we selected one node per PlanetLab site, resulting in totally 79 nodes. Each of these nodes represents a multicast group member.

2) For the 79 selected nodes Scribe and MCAN without topology awareness have been implemented. One node has been designated as a source. Among the neighbors on the overlay, *traceroutes* [95] were collected on November 10th. The hopcounts of MCAN, Scribe, multicast and m-unicast have been computed.

3) Based on the *traceroutes* collected among these 79 nodes, the underlying router-level topology has been created. This topology consisted of 4226 nodes and 7171 links. No alias resolution technique has been implemented.

4) With the knowledge of this topology, topology- aware MCAN and Scribe have been created and implemented, and subsequently, hopcount has been computed.

Figure 7.16 shows the results of our PlanetLab experiments. Figure 7.16(a) gives the pdf of hopcount in multiple MCAN1 and MCAN2. This figure reveals the same phenomenon in hopcount of MCAN1 as we observed for simulations of multiple CAN: the clustering effect. In Figure 7.16(b) we have plotted the average values of hopcount for each of the schemes. This figure resembles the corresponding Figure 7.13 and 7.15 obtained from simulation data. Again, as expected, IP multicast achieves the lowest hopcount. The values of hopcount of MCAN and Scribe (without topology awareness), seem all to be approximately equal to unicast. The hopcount of Scribe_top (with absolute topology awareness) is only slightly lower than that of MCAN2top, but lower than unicast. Striking is the hopcount of topology aware MCAN1-the hopcount of topology aware MCAN1 is even larger than that of unicast! One possible explanation for this phenomenon is illustrated in Figure 7.17. The sequence of Figures 7.17(a) to 7.17(e) shows the process of nodes joining CAN network. First, only node 1 is in CAN, and then, one by one, other nodes join. Let us assume that node 2 is the nearest node to the newcomer 5. Node 5 then chooses node 2 as the split node. However, even though they will be neighbors in CAN, due to the way MCAN algorithm operates (as described in Section), they do not send messages to each other. Hence, the hopcount among nodes will not be diminished by the nodes vicinity on CAN. In addition, as illustrated earlier in Figure 7.7 as well, the addition of node 6 (e.g. close to node 3) may separate nodes 3 and 4, which are nearby in the underlying topology.



(a) (b)

Figure 7.16: (a) The pdfs of MCAN1 and MCAN2 in multiple CAN overlays derived from experiments on PlanetLab. (b) The average hopcount of all schemes of interest, derived from experiments on PlanetLab.

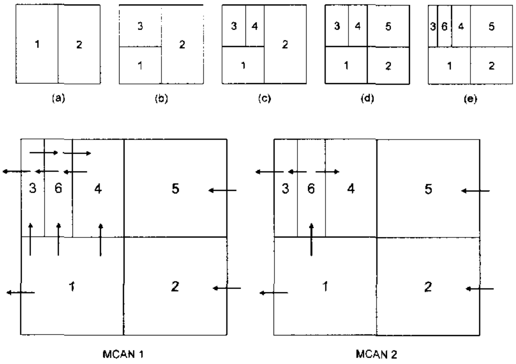


Figure 7.17: Node 5, although neighbor of node 2, does not receive messages from node 2.

7.5 Conclusions

Our goal was to examine the performance of several scalable Application Layer multicast mechanisms, MCAN and Scribe, under the same conditions, with the hopcount as a performance metric. In addition, we aimed at establishing the upper and lower boundaries for the hopcount of these two algorithms, as a function of the network awareness. Here, we summarize our observations:

1. We observe a hopcount "clustering" effect in MCAN.
2. The number of duplicate messages in MCAN is fairly eliminated with the modifications to the forwarding algorithm that we introduced.
3. There is no significant influence of topology (un)awareness on the hopcount of MCAN in scenario b (all members attached to different routers).
4. For a small number of multicast users m compared to the number of nodes in the network N ($\rho = m/N \leq 0.1$), the hopcount obtained in MCAN1 in a topology aware CAN overlay is higher than the hopcount in topology unaware CAN overlays, in scenario *b*. The same phenomenon has been observed in the measurements results. One possible explanation is the way the topological information has been integrated in CAN.
5. In Scribe, we observe a large influence of topology awareness on the hopcount. The difference in hopcount can reach 40%.
6. In scenario a, topology unaware Scribe performs worse than m-Unicast.
7. The hopcount in the topology unaware NICE is virtually the same as in m-Unicast, irrespective of the value of the cluster size l.
8. When the underlying topology is completely known, and each user is attached to a different router, Scribe achieves the same number of hops as IP multicast. The hopcount of a complete topology aware Scribe when the users may connect to the same router remains higher than that of IP multicast. MCAN seems to outperform m-Unicast only if CAN is completely topology-aware, and users may attach to the same router multiple times. Under all the other circumstances, the number of hops of MCAN is higher than m-Unicast. The modified CAN algorithm MCAN2 outperforms m-Unicast. It outperforms the topology unaware Scribe as well.

9. The results of experiments on PlanetLab match our simulation results.

   We can conclude that all the schemes perform poorly when the underlying topology is unknown. However, in case of CAN, the topology awareness can even lead to an increase in the hopcount.

Chapter 8

Conclusions

This chapter summarizes our research on Network and Application Layer multicast. We recapitulate the methods we have used and our major findings. The chapter is organized in three sections: Section 8.1 and Section 8.2 outline our findings related to Network Layer (IP) and Application Layer (AL) multicast respectively. The main contributions of this thesis are highlighted in Section 8.3.

8.1 Network Layer Multicast

Although a standard feature on most routers for a couple of years, and in spite of the growing demand, IP multicast has not been widely utilized yet. This thesis focussed on the impediments and challenges of multicast deployment. Most network providers simply disable the multicast feature because they lack the necessary knowledge to operate and manage it and they lack a business incentive. Network operators will only put effort in implementing and managing IP multicast if doing so yields (significant) financial benefits.

In order to establish a reliable multicast business framework, first all the factors that impact multicast savings need to be quantified, as well as the additional cost multicast induces. This, in turn, can only be achieved by fully understanding the behavior and the properties of multicast trees. Understanding and modeling multicast trees properly has therefore been the focus of the first part of this thesis.

We have started with the analysis of the stability of a multicast routing tree. The stability of a multicast routing tree is an important factor that contributes to the additional complexity of multicast over unicast. As multicast users join or leave the group, a routing tree has to be recomputed, which can lead to transient routing behavior and packet losses. We have examined the stability of such a tree, specifically, how the number of links changes as the number of multicast users in a group changes. In particular, the probability density function (pdf) for the number of changed links when one multicast user joins or leaves the group has been studied. We have first analyzed the class of Shortest Path Trees (SPT), since they are commonly used by the current multicast protocols. In random graphs of the class $G_p(N)$ with N nodes, link density p and with uniformly (or exponentially) distributed link weights, SPT can be asymptotically modelled with the Uniform Recursive Tree (URT). For this reason, URT has been used further as a model for multicast trees. For the above defined class of graphs the probability density function of the number of changed links is proved to tend to a Poisson distribution for large N. The proof of this theorem enables a generalization to an arbitrary topology. Extensive simulations, mainly conducted to quantify the validity of the asymptotic regime, reveal that the Poisson law seems more widely valid than just in the asymptotic regime.

Moreover, the stability of a Steiner Tree connecting m multicast users is compared to the Shortest Path Tree via simulations. Steiner Minimum Trees, although optimizing on the use of resources, are not deployed in multicast routing protocols, due to their complexity and alleged instability. Our simulations confirm the intuitive assumption that the stability of the Steiner Tree is in most situations worse than that of the corresponding Shortest Path Tree. Mainly because the departure or arrival of a multicast member may cause other branches to be included in the Steiner Tree (to achieve an overall minimum in the sum of the weights). Moreover, simulations reveal that not only the network size N, but also the link weight distribution, often forgotten when modeling the network, has a significant impact on the stability of Steiner Trees. If the majority of the links is differently weighted, the stability of the Steiner Tree resembles that of the SPT. The other extreme, where most link weights are equal, leads to large instabilities reflected by wild oscillations in the corresponding pdf of the number of changed links.

Hence, we conclude that the intuitive assumption on the (in)stability of Steiner Minimum Trees is correct. Even though Steiner Trees optimize the use of resources, they cannot be used in multicast protocols.

If we define the cost of multicast trees as the sum of used resources, then a question that arises is, how much more costly are the Shortest Path Trees compared to Steiner Trees. With the link weights representing the available resources, the sum of used resources can be expressed as the sum of the link weights in the tree. The sum of the link weights in the SPT is on average not more than 37% worse than that in SMT. From the extensive simulations we were lead to conjecture that the probability density function of the scaled sum of the link weights for small values of N tends to a Gumbel, however, with the increase of N, it converges slowly toward a Gaussian.

The possible implications for business scenarios for ISPs of the model for multicast trees we proposed have been discussed as well. Among the additional costs of multicast over cost of unicast we distinguish the deployment, management and network costs. Based on our model we have computed the network costs, and we have estimated the deployment and management ones. The results suggested that at moderate bandwidths multicast becomes beneficial for network operators for approximately one hundred receivers. For higher bandwidths the break-even number of receivers is achieved with approximately ten receivers.

In our study, we have assumed that routing follows shortest paths from the source toward each of the destinations. Policy-based routing, although designed to find short paths, can have the inflating effect on the paths. In order to convince the network operators that the business model based on the framework that we propose is trustworthy, we have investigated how well our theoretical results match data collected from the Internet itself. Specifically, we have investigated the node degree distributions and the number of links in the multicast routing trees.

Internet measurement data in our study has been collected via the traceroute utility. Traceroute collects and returns the information on an end-to-end IP path by transmitting and receiving probe packets. By merging the collected paths together, multicast routing trees can be obtained. Subsequently, the grouping of multicast routing trees can lead to maps of (a part of) the Internet on a router-level. The traceroute data that we have used was gathered from several different measurement architectures, such as CAIDA, RIPE and PlanetLab.

After analyzing this data, we have shown that traceroutes suffer from several types of drawbacks and flaws. We have demonstrated and discussed the weaknesses of traceroutes that include errors and inaccuracies, aliasing problem and the bias sampling problem. These drawbacks of traceroutes must be taken into account when drawing the conclusions on topological properties of Internet based on traceroute measurement data.

The (un)reliability of traceroutes is reflected in the results for the node degree distribution in maps of the Internet. In addition to our own measurements, measurement data for this study has been provided by a number of different research groups. Similarly to previously reported work on the node degree in the Internet map, most of the measurements we collected, with the exception of RIPE and PlanetLab data, indicate power-laws for node degrees, with similar values for the coefficient α_τ, (α_τ in the degree distribution given in Chapter 2). Furthermore, the measurement data seem to suggest that alias-resolving techniques do not have a major effect on the node degree distribution in Internet. The coefficient α_τ lies in the range $2.3 - 2.4$, with the exception of CAIDA data, where the value of the slope coefficient is $\alpha_\tau = 2.97$. This might be the consequence of the one order of magnitude larger map obtained with the Skitter project (CAIDA), than the one obtained within the Rocketfuel project. The results obtained by RIPE and PlanetLab indicate however not polynomially, but exponentially distributed node degrees.

The divergence in results obtained by the RIPE and PlanetLab may be caused by RIPE and PlanetLab measurement architecture, where each measurement box serves both as the source and as the destination. Hence, the number of sources and destinations is balanced, whereas in all the other measurement architectures the number of sources has been very limited (only few) compared to a large number of destinations. The discrepancy in the node degree distributions based on RIPE and PlanetLab traceroute data with results obtained by others, seems to confirm the above stated observation on the reliability of traceroutes.

Only few measurement results have been reported on the topology of multicast trees to this end. The results on the node degree distributions in multicast trees have also been controversial: both exponential and polynomial distribution (power-law) have been claimed. These conclusions have been drawn based on the quality of the fits of the node degree distribution with a linear function on a log-log and a log-lin scale

While for larger values of the number of destinations m the degree distribution seems to follow a power-law, for small m ($m \leq 50$) (based on both RIPE and CAIDA traceroute data) this seems not to be the case. Further, our measurement results on the number of links in multicast trees indicated that this number lies in the range $E\left[H_N\left(m\right)\right] \pm 3\sigma_N(m)$ (where $\sigma_N(m) = \sqrt{var\left(H_N\left(m\right)\right)}$, and $E\left[H_N\left(m\right)\right]$ and $var\left(H_N\left(m\right)\right)$ are the average and the variance of the number of links in the Shortest Path Tree in the random graph, and are given with Equations 2.7 and 2.9) for all values of number of users m. Additionally, since $\sigma_N\left(m\right)$ is much smaller than $E\left[H_N\left(m\right)\right]$ for large N, as also supported by the measurement data, the number of links in the tree is well approximated by the mean, $H_N\left(m\right) \approx E\left[H_N\left(m\right)\right]$ .This leads us to believe that although the random graph with uniformly (or equivalently exponentially) distributed link weights and binomially distributed node degrees is not a good model for the Internet, the Shortest Path Tree deduced from that class might be a good first-order approximation for the multicast trees on the Internet.

8.2 Application Layer Multicast

Application Layer (AL) multicast was born as a response to a slowly deployed but highly demanded IP multicast. Within a short period of time, a wealth of algorithms emerged. Whereas the initially proposed schemes suffered from scalability, recently proposed mechanisms for AL multicast can scale to large groups, counting tens of thousands of users. The low implementation barrier of AL multicast is certainly its most attractive feature. Unfortunately, the efficiency of AL multicast is lower than that of IP multicast, since packets may traverse the same link several times.

In our study on Application Layer multicast, we have investigated the total number of hops (hopcount) the message traverses in three schemes: CAN-based multicast (MCAN), Scribe and NICE. Among all the AL multicast protocols we have chosen these three due to their superior scalability over the other schemes. We have compared the hopcount among the different schemes, as well as to the hopcount achieved by unicast and IP multicast, all under equal conditions. To the best of our knowledge, no such evaluation of AL multicast has been performed to this end. Moreover, we have investigated the influence of topology awareness on the efficiency of these protocols. Topology awareness can be defined as the degree of the congruency of the overlay topology with the underlying topology. We have assumed two extreme situations: in one extreme, overlay networks are completely oblivious to the underlying networks. In the other, overlay nodes possess all the knowledge of the underlying substrate.

In addition, we have introduced improvements to the MCAN algorithm that lead to reduction of duplicate messages.

Both extensive simulations, as well as the measurements via PlanetLab, have indicated that:

- The values of hopcount of the original MCAN forwarding algorithm concentrate around several dominant values. This suggests that there seems to exist a finite number of "groups" of CAN structures.
- The number of duplicate messages in MCAN is fairly eliminated with the modifications to the forwarding algorithm that we introduced.
- Topology (un)awareness leads to no significant improvement on the hopcount of MCAN if all members can be attached to different users. For small number of users, the hopcount can even deteriorate. In Scribe, a large influence of topology awareness on the hopcount is perceived. The difference in hopcount can reach 40%.
- The original MCAN forwarding algorithm seems to outperform unicast only if CAN overlay is completely topology aware and multiple users may attach to the same router. Under all the other circumstances, the number of hops of MCAN is higher than that obtained with unicast. Our modified MCAN algorithm outperforms unicast under all circumstances.
- If multiple users may attach to a same router, topology unaware Scribe performs worse than m-Unicast.
- The hopcount in the topology unaware NICE is virtually the same as in m-Unicast, irrespective of the value of the cluster size l.
- When the underlying topology is completely known and each user is attached to a different router, Scribe achieves the same number of hops as IP multicast. The hopcount of a complete topology aware Scribe when the users may connect to the same router is still higher than that of IP multicast.

8.3 Thesis Contributions

Below, we list the main contributions of this thesis.

- We proposed a Uniform Recursive Tree (URT) as a model for IP multicast trees. Based on the URT, the stability, savings and cost of multicast trees have been analyzed and quantified.

- The proposed model (URT) has been verified via measurements on the Internet. The measurements of the number of links of multicast trees in the Internet indicate an excellent fit with our analytical derivations. This suggests that the Shortest Path Tree derived from the random graph is a good approximation for multicast trees, even though the random graph is not a good model for the Internet itself.

- The possible implications of our theoretical model for business scenarios of the Internet Service Providers have been investigated. No appropriate business model for IP multicast has been established to this end. Our results suggested that at moderate bandwidths multicast becomes beneficial for network operators for approximately one hundred receivers. For higher bandwidths the break-even number of receivers is achieved with approximately ten receivers.

- The node degree distribution in the map of the Internet and in the IP multicast trees, both constructed based on *traceroute* measurements collected in various measurement architectures, has been analyzed. So far, power-laws have been claimed for these distributions. The majority of our results for the node degree distributions in the map of the Internet indicates power-laws, with a variant gradient coefficient. Nevertheless, in some architectures (RIPE and PlanetLab) we observe exponentially and not polynomially distributed degrees. Similar observations have been obtained for multicast trees. While for larger values of the number of multicast receivers the degree distribution seems to follow a power-law, for small group sizes this seems not to be the case.

- Several scalable AL multicast schemes have been evaluated (under the same conditions) and the influence of the topology awareness has been evaluated. To the best of our knowledge, no such analysis has been conducted previously. We may conclude that, among the schemes evaluated, Scribe performs the best. When the topology is unknown to the end user, the performance of these schemes (in terms of the number of traversed hops) can significantly deteriorate, and fall behind that of unicast. In addition, the creation of CAN overlays indicates that the method of the topology-information integration is even more important than the integration of the topology information into the overlay itself.

Chapter 9

Abbreviations

AL	Application Layer
ALMI	Application Level Multicast Infrastructure
ARPA	Advanced Research Project Agency
AS	Autonomous System
ASM	Any Source Multicast
BGMP	Border Gateway Multicast Protocol
BGP	Border Gateway Protocol
BTP	Banana Tree Protocol
CAIDA	Cooperative Association for Internet Data Analysis
CAN	Content Addressable Network
DHT	Distributed Hash Table
DIS	Distributed Interactive Simulations
DT	Delaunay Triangulation
DVMRP	Distance Vector Multicast Routing Protocol
HMTP	Host Multcast Tree Protocol
HTML	Hyper-Text Markup Language
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IGMP	Internet Group Membership Protocol
IP	Internet Protocol
ISP	Internet Service Provider
MASC	Multicast Address-Set Claim
MBGP	Multiprotocol Border Gateway Protocol
MBone	Multicast Backbone
MCAN	CAN-based multicast
MSDP	Multicast Source Discovery Protocol
MST	Minimum Spanning Tree
NCP	Network Communication Protocol
NICE	NICE is the Internet Cooperative Environment

p2p	Peer to Peer
pdf	Probability density function
PIM-DM	Protocol Independent Multicast-Dense Mode
PIM-SM	Protocol Independent Multicast-Sparse Mode
RFC	Request for Comments
RGU	Random Graph with uniformly distributed link weights
RIPE	Réseaux IP Européen
RP	Rendezvous Point
RPF	Reverse Path Forwarding
rtt	round trip time
SA	Source Active
SCX	Scattercast Proxies
SMT	Steiner Minimum Tree
SPT	Shortest Path Tree
SSM	Source Specific Multicast
TBCP	Tree Building Control Protocol
TCP	Transmission Control Protocol
TTL	Time to Live
URT	Uniform Recursive Tree
VLAN	Virtual Local Area Network

Chapter 10

Symbols

α	power exponent in polynomial link weight distribution
α_τ	power exponent in the node degree distribution
$\Gamma(z)$	Gamma function
γ	Euler's constant
$\Delta_N(m)$	the number of changed links in the tree
$\delta(v)$	"attribute" of node v
ν	the economy-of-scale factor in Chuang-Sirbu law
$\pi(v)$	predecessor list of node v
$\psi(x)$	digamma function
$\zeta(z)$	Riemann Zeta function

C	cost
$\deg_n$	node degree
$E[H_N(m)] = f_N(m)$	the average number of links in a tree
$E[H_N(m)] = g_N(m)$	multicast efficiency
$G_p(N)$	a random graph with N nodes and link probabilty p
H_N	the number of links in a path from a source to a destination (hopcou
$H_N(m)$	the number of links in a tree from a source to m destinations
K_N	a complete graph with N nodes
L	the number of links in the graph
l	the cluster size in NICE
m	the number of multicast receivers
m_t	the number of multicast users (receivers plus source)
N	the number of nodes in the graph/ network size
N_{Pastry}	the number of Pastry users
r	correlation coefficient
$w(i \to j)$	weight on the link connecting node i to node j
$W_N(m)$	the sum of the link weights in SPT
$W_{Steiner}(k)$	the sum of the link weights in SMT

Bibliography

[1] http://dast.nlanr.net/projects/beacon/.

[2] http://www.caida.org.

[3] http://www.multicasttech.com/status/.

[4] http://www.planet-lab.org/.

[5] Ripe test traffic measurements. *http://www.ripe.net/ripencc/mem-services/ttm/.*

[6] D3.5 deliverable: the cost of multicast compared to unicast. April 2003.

[7] M. Abramowitz and I.A. Stegun. *Handbook of Mathematical Functions.* Dover, 1968.

[8] K.C. Almeroth. Evolution of multicast: From the MBone to interdomain multicast to Internet 2 deployment. *IEEE Network*, 14(8):10–20, January/February 2000.

[9] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient overlay networks. *Proc. of 18th ACM Symposium on Operating Systens Principles (SOSP)*, pages 131–145, October 2001.

[10] H. Asaeda. Protocol analysis of any-source multicast and source-specific multicast. *INRIA Research Report, RR-5080*, January 2004.

[11] S. Banarjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. *Proc. of ACM SIGCOMM*, pages 205–217, August 2002.

[12] S. Banarjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller. Construction of an efficient overlay multicast infrastructure for real-time applications. *Proc. of IEEE INFOCOM'03*, April 2003.

[13] T. Bates, Y. Rekhter, R. Chandra, and D. Katz. Multiprotocol extensions for BGP-4. *RFC 2858*, June 2000.

[14] R. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, (16):87–90, 1958.

[15] S. Bhattacharyya. An overview of source-specific multicast (SSM). *RFC 3569*, July 2003.

[16] S. Birrer and F. E. Bustamante. Resilient peer-to-peer multicast from the ground up. *Proc. of IEEE Network Computing and Applications (NCA'04)- Workshop on Adaptive grid Computing*, August-September 2004.

[17] R. Boivie, N. Feldman, Y. Imai, W. Livens, D. Ooms, and O. Paridaens. Explicit multicast (Xcast) basic specifications. *IETF Internet Draft, <draft-ooms-xcast-basic-spec-03.txt>*, June 2002.

[18] B. Bollobas. *Random Graphs.* Academic Press, 1985.

[19] B. Bollobas, D. Gamarnik, O. Riordan, and B. Sudako. On the value of a random minimum weight Steiner tree. *Combinatorica*, 24(2):187–207, April 2004.

[20] H. Burch and B. Cheswick. Mapping the Internet. *IEEE Computer*, 32(4):97–98, 102, April 1999.

[21] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan. Internet group membership protocol, version 3. *RFC 3376*, October 2002.

[22] M. Castro, M. Costa, and A. Rowstron. Should we build gnutella on a structured overlay? *ACM SIGCOMM Computer Communication Review*, 34(1):131–136, January 2004.

[23] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron. Topology-aware routing in structured peer-to-peer overlay networks. *Technical Report Microsoft Research MSR-TR-2002-82*, 2002.

[24] M. Castro, M. Jones, A-M Kermarrec, A. Rowstron, M. Theimer, H. Wang, and A. Wolman. An evaluation of scalable application-level multicast using peer-to-peer overlays. *Proc. of IEEE INFOCOM'03*, pages 1510–1520, March-April 2003.

[25] R. Chalmers and K. Almeroth. On the topology of multicast trees. *IEEE/ACM Transactions on Networking*, 11(1):153– 165, February 2003.

[26] R. C. Chalmers and K. C. Almerorth. Modeling the branching characteristics and efficiency gains in global multicast trees. *IEEE INFOCOM2001, Alaska*, 2001.

[27] Y. Chawathe. Scattercast: an adaptable broadcast distribution framework. *Multimedia Systems*, 9(3):104–118, July 2003.

[28] Y. Chawathe, S. McCanne, and E. Brewer. Rmx: Reliable multicast for heterogeneous networks. *Proc. of IEEE INFOCOM'00, Tel Aviv, Israel*, pages 795–804, March 2000.

[29] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making gnutella-like p2p systems scalable. *Proc. of ACM SIGCOMM'03*, Augustus 2003.

[30] Q. Chen, H. Chang, R. Govindan, S. Jamin, S. J. Shenker, and W. Willinger. The origin of power laws in Internet topologies revisited. *Proceedings of IEEE INFOCOM'02*, pages 608–617, 2002.

[31] Y. Chu, S. G. Rao, S. Seshan, and H. Zhang. A case for end system multicast. *IEEE Journal on Selected Areas in Communications (JSAC)*, 20(8):1456–1471, October 2002.

[32] J. C. I. Chuang and M. A. Sirbu. Pricing multicast communication: A cost-based approach. *Telecommunication Systems*, 17(3):281–297, July 2001.

[33] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *An Introduction to Algorithms.* MIT Press, Boston, 2000.

[34] S. Deering. Host extensions for IP multicasting. *RFC 1112*, August 1989.

[35] S. Deering. *Multicast routing in a datagram internetwork.* Stanford University, Stanford, CA, USA, 1992.

[36] S. Deering, D. L. Estrin, D. Farinacci, V. Jacobson, A. Helmy, D. Meyer, and L. Wei. Protocol Independent Multicast version 2 dense mode specification. *Internet Draft, Internet Engineering Task Force, Work in progress*, June 1999.

[37] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, (1):269–271, 1959.

[38] M. Doar and I. Leslie. How bad is naive multicast routing? *Proc. of IEEE INFOCOM'93*, pages 82–89, April 1993.

[39] D. Dolev, O. Mokryn, and Y. Shavitt. On multicast trees: Structure and size estimation. *Proc. of IEEE INFOCOM'03*, 4(2):1011–1021, March 30 - April 3 2003.

[40] P. Druschel, M. Castro, A.-M. Kermarrec, and A. Rowstron. Scribe: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications (JSAC)*, 20(8):1489–1499, October 2002.

[41] B. Edwards, L. Giuliano, and B. Wright. *Interdomain Multicast Routing.* Addison Wesley, May 2002.

[42] H. Einsiedler, P. Hurley, B. Stiller, and T. Braun. Charging multicast communications based on a tree metric. *Proc. of Multicast Workshop, Braunschweig, Germany,* 1999.

[43] H. Eriksson. MBone: the multicast backbone. *Communications of the ACM,* 37(8):54–60, 1994.

[44] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei. Protocol independent multicast (PIM-SM): Protocol specifications. *RFC 2362,* June 1998.

[45] P. Winter F. Hwang, D. Richards. *The Steiner Tree Problem,* volume 53. (Annals of Discrete Mathematics), North Holland, 1992.

[46] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the Internet Topology. *Proceedings of ACM SIGCOMM'99, Cambridge, Massachusetts,* pages 251–262, 1999.

[47] B. Fenner and D. Meyer. Multicast source discovery protocol (MSDP). *RFC 3618,* October 2003.

[48] W. Fenner. Internet group membership protocol, version 2. *RFC 2236,* November 1997.

[49] W. Fenner and S. Casner. A 'traceroute' facility for IP multicast. *Tech. Rep, draft-ietf-idmr-traceroute-ipm-*txt, Internet Engineering Task Force (IETF),* August 1998.

[50] P. Francis. Yoid: Extending the Internet multicast architecture. *Unrefereed report, http://www.icir.org/yoid/docs/index.html,* pages 1–38, April 2000.

[51] Freenet. *http://freenet.sourceforge.net,* 2003.

[52] R. Govindan and H. Tangmunarunkit. Heuristics for Internet map discovery. *Proc. of IEEE INFOCOM'00,* pages 1371–1380, March 2000.

[53] P. K. Gummadi, R.J. Dunn, S. Saroiu, S.D. Gribble, H.M. Levy, and J. Zahorjan. Measurement, modeling and analysis of a peer-to-peer file-sharing workload. *Proc. of SOSP'03,* October 2003.

[54] F. Harary. *Graph Theory.* Addison-Wesly Publishing Company, Reading, Massachusetts, 1969.

[55] R. Hekmat and P. Van Mieghem. Degree distribution and hopcount in wireless adhoc networks. *Proc. of 11th IEEE International Conference on Networks (ICON 2003)*, Sept.28-Oct.1 2003.

[56] D. Helder and S. Jamin. End-host multicast communication using switch-trees protocols. *Proc. of the 2nd IEEE/ACM International Symposium on Cluster Computing and Grid (CCGRID'02)*, May 2002.

[57] S. Herzog, S. Shenker, and D. Estrin. Sharing the 'cost' of multicast trees: An axiomatic analysis. *IEEE/ACM Transactions on Networking*, 5(6):847–860, December 1997.

[58] H. Holbrook and B. Cain. Source-specific multicast for IP. *Internet-Drafts <draft-ietf-ssm-arch-06.txt>*, September 2004.

[59] H. Holbrook and D. Cheriton. IP multicast channels: Express support for large-scale single-source applications. *Proc. of ACM SIGCOMM'99*, pages 65–78, August 1999.

[60] Christian Huitema. *Routing on the Internet.* Prentice Hall PTR, New Jersey, 1995.

[61] S. Jain, R. Mahajain, D. Wetherall, and G. Borriello. Scalable self-organizing overlays. *Technical Report University of Washington UW-CSE 02-02-02*, February 2002.

[62] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, and J. O'Toole. Overcast: Reliable multicasting with an overlay network. *Proc. of USENIX*, October 2000.

[63] S. Janson, T. Luczak, and A. Rucinski. *Random Graphs.* Wiley & Sons, New York, 2000.

[64] L.R. Ford (Jr). Network flow theory. *The RAND Corporation, Santa Monica, California*, page 293, August 14 1956.

[65] R. Karp. Reducibility among combinatorial problems. *Complexity of Computer Communications*, pages 85–103, 1972.

[66] E. Kranakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. *Proc. of 11th Canadian Conference on Computational Geometry (CCCG'99)*, pages 51–54, August 1999.

[67] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proc. of the American Mathematical Society*, 7:48–50, 1956.

[68] M. Kwon and S. Fahmy. Topology-aware overlay networks for group communication. *Proc. of ACM NOSSDAV'02*, pages 127–136, May 2002.

[69] A. Lakhina, J. Byers, M. Crovella, and P. Xie. Sampling biases in IP topology measurements. *Proc. of IEEE INFOCOM'03*, pages 332–341, March 30 - April 3 2003.

[70] J. Liebeherr and T. K. Beam. Hypercast: A protocol for maintaining multicast group members in a logical hypercube topology. *Proc. of the First International Workshop on Networked Group Communication (NGC '99)*, 1736:72–89, July 1999.

[71] J. Liebeherr, M. Nahas, and W. Si. Application-layer multicasting with delaunay triangulation overlays. *IEEE Journal on Selected Areas in Communications (JSAC)*, 20(8):1472–1488, October 2002.

[72] MBONED mailing list. http://antc.uoregon.edu/mboned/.

[73] L. Mathy, R. Canonico, and D. Hutchinson. An overlay tree building control protocol. *Proc. of 3.rd International COST264 Workshop on Networked Group Communication (NGC'01)*, pages 78–87, November 2001.

[74] D. Meyer and P. Lothberg. GLOP addressing in 233/8. *RFC 3180*, September 2001.

[75] P. Van Mieghem and S. van Langen. Influence of the link weight structure on the shortest path. *to appear in Physical Review E*, 2005.

[76] J. Pansiot and D. Grad. On routes and multicast trees in the Internet. *ACM Computer Communication Review*, 28(1):41–50, January 1998.

[77] V. Paxson. End-to-end Internet packet dynamic. *Proc. of the ACM SIGCOMM'97*, October 1997.

[78] V. Paxson. End-to-end routing behavior in the Internet. *IEEE/ACM Transactions on Networking*, 5(5):601–615, October 1997.

[79] V. Paxson. End-to-end Internet packet dynamics. *IEEE/ACM Transactions on Networking*, 7(3):277–292, June 1999.

[80] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel. ALMI: An application level multicast infrastructure. *Proc. of 3rd Usenix Symposium on Internet Technologies and Systems (USITS)*, pages 153–162, March 2001.

[81] G. Phillips, S. Schenker, and H. Tangmunarunkit. Scaling of multicast trees: Comments on the Chuang-Sirbu scaling law. *ACM Sigcomm99*, 1999.

[82] R. C. Prim. Shortest connection networks and some generalizations. *Bell Systems Technical Journal*, 36:1389–1401, 1957.

[83] P. Radoslavov, D. Estrin, M. Handley, S. Kumar, and D. Thaler. The multicast address-set claim (MASC) protocol. *RFC 2909*, September 2000.

[84] Maria Ramalho. Intra-and inter-domain multicast routing protocols: A survey and taxonomy. *IEEE Communications Surveys and Tutorials*, 3(1), 2000.

[85] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. *Proc. of ACM SIGCOMM'01*, pages 161–172, August 2001.

[86] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Application-level multicast using content-addressable networks. *Proc. of the 3rd International Workshop on Networked Group Communication (NGC '01)*, 2001.

[87] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-aware overlay construction and server selection. *Proc. of IEEE INFOCOM'02*, June 2002.

[88] V. Roca and A. El-Sayed. A host-based multicast (HBM) solution for group communications. *Proc. of 1st International Conference on Networking (ICN'01)*, pages 610–619, 2001.

[89] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. *Proc. of 18th IFIP/ACM Conference on Distributed Systems Platforms*, November 2001.

[90] M. Hofmann S. K. Kasera and R. E. Miller. A profitable multicast business model. *to appear in Computer Communications Journal.*

[91] R. Sibson. Locally equiangular triangulations. *The Computer Journal*, 21(3):243–245, 1977.

[92] R. T. Smythe and H. M. Mahmoud. A survey of recursive trees. *Theory of Probability and Mathematical Statistics*, (51):1–27, 1995.

[93] The Gnutella Protocol specification. *http://dss.clip2.com/GnutellaProtocol104.pdf*, 2000.

[94] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. Measuring ISP topologies with rocketfuel. *IEEE/ACM Transactions on Networking*, 12(1):2–16, February 2004.

[95] W. Richard Stevens. *TCP/IP Illustrated*, volume 1, The Protocols. Addison-Wesley, Reading, Massachusetts, 1994.

[96] E. Takahashi, Y. Tanaka, T. Ohara, and T. Miyoshi. Cost-based pricing for multcast streaming services. *Proc. of 10th International Telecommunication Network Strategy and Planning Symposium*, pages 245–250, June 2002.

[97] D. Thaler. Border gateway multicast protocol (bgmp): Protocol specification. *RFC 3913*, September 2004.

[98] D. Tran, K. Hua, and T. Do. ZIGZAG: an efficient peer-to-peer scheme for media streaming. *Proc. of IEEE INFOCOM'03*, March-April 2003.

[99] R. van der Hofstad, G. Hooghiemstra, and P. Van Mieghem. Size and weight of shortest path trees with exponential link weights. *to appear in Combinatorics, Probability and Computing*, 2005.

[100] P. Van Mieghem, G. Hooghiemstra, and R. van der Hofstad. A scaling law for the hopcount in Internet. *Technical Report 2000125, Delft University of Technology, http://www.nas.ewi.tudelft.nl/people/piet/telconference.html*, 2000.

[101] P. Van Mieghem, G. Hooghiemstra, and R. van der Hofstad. On the efficiency of multicast. *IEEE/ACM Transactions on Networking*, 9(6):719–732, 2001.

[102] P. Van Mieghem, G. Hooghiemstra, and R. W. van der Hofstad. Stochastic model for the number of traversed routers in Internet. *Proc. of Passive and Active Measurements (PAM2001)*, pages 190–194, April 2001.

[103] D. Waitzman, C. Partridge, and S. Deering. Distance vector multicast routing protocol. *RFC 1075*, November 1988.

[104] M. Waldvogel and R. Rinaldi. Efficient topology-aware overlay network. *ACM SIGCOMM Computer Communications Review*, 33(1):101–106, January 2003.

[105] L. Wei and D. L. Estrin. The trade-offs of multicast trees and algorithms. *Proc. of the 3rd International Conference on Computer Comunications and Networking (ICCCN'94)*, pages 17–24, September 1994.

[106] B. Yao, R.Viswanathan, F. Chang, and D. Waddington. Topology inference in the presence of anonymous routers. *Proc. of IEEE INFOCOM'03*, pages 353–363, March 30 - April 3 2003.

[107] D. Zappala and A. Fabbri. Using ssm proxies to provide efficient multiple-source multicast delivery. *Proc. of IEEE Globecom'01*, November 2001.

[108] E. W. Zegura, K. L. Calvert, and M. J. Donahoo. A quantitative comparison of graph-based models for Internet topology. *IEEE/ACM Transactions on Networking*, 5(6):770–783, December 1997.

[109] B. Zhang, S. Jamin, and L. Zhang. Host multicast: A framework for delivering multicast to end users. *Proc. of IEEE INFOCOM'02*, pages 1366–1375, June 2002.

[110] R. Zhang and Y. C. Hu. Borg: A hybrid protocol for scalable application-level multicast in peer-to-peer networks. *Proc. of ACM NOSSDAV'03*, June 2003.

[111] Y. B. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for faulttolerant wide-area location and routing. *Technical Report UCB/CSD-01-1141, UC Berkeley*, April 2001.

[112] S. Q. Zhuang, B. Y. Zhao, and A. D. Joseph. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. *Proceedings of 11th ACM/IEEE NOSSDAV'01*, June 2001.

Summary

Computing systems that can treat and provide multimedia keep developing rapidly. Parallel to this development, great progresses in network technologies, in wireless as well as wired, in core as well as access networks, are achieved: high-speed broadband networks are becoming a reality. The continuously increasing number of users in the Internet contributes to creating good foundations for the distribution of high quality multimedia content (audio/video conferencing, real-time interactive applications, shared white boards, software updates, tele-classing, animated simulations, etc.) to the end users.

The distribution of these types of applications over Internet can be most effectively enabled by network-layer (IP) multicast technology. IP multicast has emerged to avoid the situation in which multiple packets with identical data are sent to multiple receivers. IP multicast differs from unicast in that it enables the source of data to send the data packet only once. That packet is duplicated in the network when needed. In this way multicast induces considerable reductions in the load on the sender, as well as the overall network load.

Despite the savings it offers in network capacity and the growing demand for the applications it supports, IP multicast is fifteen years after its origination still only restrictedly adopted. The reasons that lead to its slow deployment are discussed in Chapter 3. One of the most important ones is the lack of an appropriate business model. The network operators are reluctant to implement multicast due to its higher complexity compared to unicast. Such a business model can only be formulated if all the factors that impact the savings and additional costs of multicast over unicast can be quantified. This in turn requires the properties and behavior of multicast routing trees to be understood and determined.

Consequently, the first part of this thesis focusses on determining the representative properties of multicast routing trees. This is achieved both through exercising graph theory (Chapter 4) as via measurements on the Internet (Chapter 5). The properties analytically studied in Chapter 4 are the number of links (branches) in a multicast tree, the stability of a tree, and its cost, defined as the sum of the used resources. They are mathematically derived for the Uniform Recursive Tree, that serves as a model for multicast trees. The possible implications of our results for network operators are discussed in the same chapter.

In addition to the study of Internet multicast trees, Chapter 5 presents the study of modeling (a part of) the Internet. In both studies we make use of different measurement architectures. The Internet measurements are performed via the *traceroute* utility. *Traceroute* represents the current best tool for retrieving the path between a sender and the destination in the Internet. Nevertheless, it suffers from several types of flaws (discussed in the same chapter). Regarding multicast, the Internet measurements we have collected seem to indicate that the Uniform Recursive Tree represents a reasonable first-order approximation for multicast trees in the Internet.

Recently, an alternative solution to the IP multicast deployment problems emerged, in the form of Application Layer (AL) multicast. The major difference between IP multicast and AL multicast is that in the former, packets are replicated at routers, whereas in the latter, packets are replicated at end-users. End-users that want to receive multicast traffic form an overlay network in which each edge corresponds to a direct unicast path between two group members. All data packets are sent as unicast packets and forwarded from one member to another on the overlay, according to a particular algorithm. The price that AL multicast has to pay for its simplicity is a performance penalty (e.g. in terms of number of hops and packet delay), because a packet may be replicated and forwarded on the same link more than once. Due to the ease of deployment, many AL multicast protocols have been proposed within a couple of years. However, little is known about their efficiency thus far. In general, it is difficult to compare the efficiency for most protocols, as the performance also varies according to group sizes and the characteristics of the underlying topology. In most cases different algorithms are not compared under the same conditions. Understanding the performance of existing AL multicast protocols is invaluable for optimizing and improving the best among them. Moreover, if AL multicast does not substantially improve unicast, its deployment cannot be justified.

Therefore, in the second part of this thesis we aim to compare AL multicast protocols under equal conditions, as a function of the network size and the number of multicast users. Chapter 6 reviews seventeen different AL multicast protocols. Among those, three that are most promising in terms of efficiency and scalability, have been chosen. We further introduce improvements to one of them. These four protocols have been compared mutually, as well as to IP multicast and unicast. As a comparison metric, the number of traversed hops has been chosen. In addition, the influence of the knowledge of the underlying topology has been investigated. The results of our analysis are given in Chapter 7. The extensive simulations and experiments on the Internet have suggested that when the underlying topology is unknown to the end users, and they may be attached to the same router, the efficiency of AL multicast is worse than that of unicast.

Samenvatting (Summary in Dutch)

Titel: Multicast in de netwerk- en de applicatielaag.

Multimedia toepassingen over het Internet, zoals audio- en videoconferenties, telelearen en software updates, laten een snelle groei in hun omvang zien. De oorzaak van deze groei is de toenemende rekenkracht van computers, de grote vooruitgang in netwerktechnologieën (breedband), en de toename in het aantal gebruikers.

Het over het Internet verzenden van data gebeurt met name middels het unicast communicatie protocol. Echter, het verzenden van multimedia tussen meerdere gebruikers kan efficiënter verzorgd worden door middel van het multicast protocol. Het verschil tussen unicast en multicast is het volgende: bij unicast zendt een zender een bericht aan de ontvanger via de kortst mogelijke route. In het geval van meerdere ontvangers stuurt de bron een bericht evenveel keer als er ontvangers zijn. Multicast daarintegen combineert de afzonderlijke unicast routes in een boomstructuur met de zender aan de basis en de ontvangers aan de uiteinden. Er wordt steeds slechts één bericht (pakketje) door de bron (onderaan aan de stam) gestuurd, tot op het plek waar de takken van de boom uit elkaar gaan. Op die plek wordt het bericht vermenigvuldigd. De zo ontstane multicast routeringsboom levert een forse besparing van de netwerkcapaciteit op voor multimedia toepassingen.

Ondanks de besparing van netwerkcapaciteit is IP multicast (multicast op het netwerk niveau) vijftien jaar na het ontstaan ervan nog steeds niet op grote schaal doorgebroken. Er zijn een aantal redenen die de implementatie van multicast hebben vertraagd (Hoofdstuk 3). Eén van de belangrijkste redenen is het ontbreken van een geschikt business model. De netwerkbeheerders zullen multicast pas implementeren als het winst oplevert. Een business model kan pas worden opgesteld als alle factoren die besparingen en kosten van multicast beïnvloeden kunnen worden gekwantificeerd. Dit betekent dat we de eigenschappen van multicast routeringsbomen, en meer algemeen de onderliggende Internettopologie, moeten bepalen en kwantificeren.

Als eerste doel van deze thesis stellen wij de representatieve eigenschappen van de multicast routeringsbomen vast door middel van zowel grafentheorie (Hoofdstuk 4) als Internetmetingen (Hoofdstuk 5). De eigenschappen die we in Hoofdstuk 4 analytisch bestuderen zijn het aantal links (takken) in een multicastboom, de stabiliteit van de multicastboom, en zijn kosten. Onze wiskundige berekeningen zijn gebaseerd op de

Uniforme Recursieve Boom. Met behulp hiervan hebben we een schatting gemaakt onder welke condities multicast winstgevend wordt voor netwerkoperatoren.

Naast multicastbomen bestuderen wij in Hoofdstuk 5 het modelleren van de topologie van (een deel van) het Internet. Daarvoor maken wij gebruik van verschillende meetopstellingen en architecturen. De Internet metingen zijn met name verricht door middel van de '*traceroute*' methode. *Traceroute* is op dit moment het beste gereedschap voor het bepalen van het pad van de zender naar de ontvanger. *Traceroute* is echter niet ideaal vanwege de aanwezigheid van foutmeldingen, aliasing en een bias sampling probleem (Hoofdstuk 5). De effecten van deze nadelen moeten worden meegewogen in conclusies over de Internettopologie en routeringsbomen.

Onze metingen aan het Internet suggereren dat de Uniforme Recursieve Boom een aanvaardbaar model is voor multicastbomen in het Internet. Dit model kan daarom als basis dienen voor een multicast business model.

De problemen die inherent zijn aan IP multicast kunnen deels opgelost worden met Application Layer (AL) multicast - multicast op de applicatie laag. In AL-multicast is de multicastfunctionaliteit van de netwerklaag naar de applicatielaag verplaatst. Dat wil zeggen dat het dupliceren van de pakketjes niet meer door de routers, maar door de eindgebruikers wordt gedaan. Echter, dit gaat ten koste van de kennis van de onderliggende netwerkstructuur en daarom is AL-multicast minder efficiënt dan IP-multicast. Er was nog weinig bekend over de prestaties van de verschillende AL-multicast protocollen. Doorgaans is het moeilijk om de efficiëntie van de meeste protocollen te vergelijken op basis van de aanwezige literatuur, omdat de resultaten veelal door middel van simulaties dan wel empirisch verkregen zijn en alleen in grafische vorm beschikbaar zijn. Meestal zijn protocollen niet vergeleken onder dezelfde condities. Bovendien varieert de efficiëntie als functie van het aantal gebruikers en de karakteristieken van het onderliggende netwerk.

Als tweede doel heeft deze thesis daarom de prestaties van AL multicast protocollen vergeleken, als een functie van de netwerkomvang en het aantal gebruikers. Deze thesis vergelijkt de bestaande protocollen wel onder gelijke condities, waardoor de protocollen beter begrepen kunnen worden. Het biedt ook een basis om te zien of AL-multicast protocollen überhaupt significant beter zijn dan unicast en of hun installatie gerechtvaardigd is. Hoofdstuk 6 presenteert een classificatie en een gedetailleerd overzicht van zeventien verschillende AL-multicast protocollen. Uit deze zeventien protocollen hebben we de drie waarvan we de beste prestaties verwachtten, met betrekking tot schaalbaarheid en efficiëntie, verder geanalyseerd. Deze drie zijn MCAN, Scribe en NICE. Daarnaast hebben we een verbetering voor het MCAN algoritme geïntroduceerd als vierde AL multicast protocol. Deze vier protocollen plus unicast en IP-multicast hebben we vergeleken, onder gelijke condities, op basis van de hopcount (aantal verbindingen tussen knooppunten in het netwerk die een bericht moet passeren om alle ontvangers te bereiken). Daarnaast hebben we onderzocht hoe het topologiebewustzijn van de protocollen hun efficiëntie beïnvloedt. Topologiebewustzijn is gedefinieerd als de mate van overeenkomst tussen de topologie van het netwerk op de applicatielaag en de topologie van het onderliggende netwerk. De vergelijking ging uit van twee uitersten: de eindgebruikers die het overkoepelende netwerk vormen weten of niets of alles van het onderliggende netwerk. De resultaten van onze analyse presenteren wij in Hoofdstuk 7. De uitgebreide simulaties en Internet metingen demonstreren dat als de onderliggende topologie onbekend is, en meerdere gebruikers aan dezelfde router verbonden kunnen zijn, de efficiency van AL multicast slechter is dan die van unicast.

Acknowledgements

This thesis describes the work that I have performed during my studies towards Ph.D. in the Network, Architecture and Services group at the Delft University of Technology, in the Netherlands. Several people contributed to this thesis coming to be what it is. Here, I would like to thank those I feel contributed the most.

In the first place, I would like to thank my supervisor and promotor professor Piet Van Mieghem. I am thankful to him for sharing his insights, ideas and his knowledge, for his feedback and productive criticism and for his enthusiasm. Working closely with him and under his guidance has been a great learning experience in many aspects, equipping me, so I believe, with a solid basis for my future professional life.

I would like to express my gratitude to the members of my Ph.D. defense committee, for honouring me by taking part in it, for reading my manuscript and for much appreciated feedback. I am particularly indebted to Prof. Maarten van Steen, Prof. Henk Sips and Prof. Stamatis Vassiliadis, for their invaluable remarks and suggestions, which amounted to a higher quality of this thesis. I would further like to give my special thanks to Prof. Stamatis Vassiliadis, for monitoring my progress during my Ph.D. studies from nearly the beginning. I thank him for each advice and the memorable words of encouragement.

I would like to acknowledge Robert Chalmers (University of California Santa Barbara), Neil Spring (University of Washington), Andre Broido and KC Claffy (CAIDA), and Henk Uijterwaal (RIPE NCC), for providing me with their Internet measurement data.

It has been a real blessing to be surrounded on a daily basis by such wonderful people as are my colleagues and friends Fernando Kuipers, Ramin Hekmat and Stijn van Langen. I am grateful to them for our numerous scientific discussions, and I will always treasure their multi-dimensional support. I would also like to thank Xiaoming Zhou and Novi Ineke Cempaka Wangi, the students I have guided during their work on their Master Theses, for their efforts that found place in some parts of this thesis. Many thanks to all the other members of the NAS and WMC groups, for we all together created a friendly and cheerful environment, pleasurable to work in.

When embarking on an adventure of pursuing a Ph.D. degree in a foreign country on your own, leaving your family and friends behind, the significance of the friends you gain at your new home is difficult to express in words. It is hard to find the right words to thank my friend Maja Pantic. I am beholden to her for her true friendship, for her contaminating enthusiasm and energy, for providing me with a range of opportunities, both in private and professional sphere. I further want to thank Dragana, Francesco and Ivan and all my other caring friends for sharing with me both joyful and painful moments in this period of my life.

Finally, there are three special people I want to thank to. My deepest thanks go to Ronald Broeke for his immeasurable support and inspiration, for saying the right words at the right time, and for convincing me of the successful accomplishment of this journey. Last, but far from the least, I am grateful to my parents Milka and Radovan Janić. I am grateful for the investments they have made in my education, for the opportunities they have given me, for always stimulating me to learn and grow and for supporting me in all my decisions. I thank them for their unconditional love, for their genes and for being my greatest fans and friends of all.

Curriculum Vitae

Milena R. Janić was born on October 26th 1974, in Belgrade, capital of Serbia and Montenegro. She graduated from the Faculty of Electrical Engineering, University of Belgrade, with major in electronics, telecommunications and control in 1999, with average mark 9.41 (10). In period February 2000-August 2000 was affiliated with a telecommunication company "Teleinformatik-Ruraltel" in Belgrade. In August 2000, she started her work toward the Ph.D. degree at the Delft University of Technology, in the Network Architectures and Services group (faculty of Electrical Engineering, Mathematics and Computer Science), under the supervision of professor dr. ir. Piet Van Mieghem. For her research, she has been nominated for the KiVI Telecommunicatieprijs, the award issued annually by the Dutch Royal Engineers Association, for the best research in the field of telecommunications. She served as a reviewer for Computer Networks and Computer Communications Journals, IEEE Communications Letters and IEEE INFOCOM, QofIS, MIPS and CoNext conferences. During her Ph.D. studies she assisted in lecturing the Telecommunication Networks and Performance Analysis courses, and she guided 4 Master of Science students.

List of publications:

1. M. Janic and P. Van Mieghem, "*On properties of multicast routing trees*", 2005, to appear in International Journal of Communication Systems.
2. M. Janic, N. Ineke, X. Zhou and P. Van Mieghem, "*Hopcount in Application Layer Multicast Schemes*", 2005, Proc. of 4th IEEE International Symposium on Network Computing and Applications (IEEE NCA05), July 27-29, Cambridge, MA, USA.
3. M. Janic and P. Van Mieghem, 2004, "*The Gain and Cost of Multicast Routing Trees*", Proc. of IEEE International Conference on Systems, Man and Cybernetics (IEEE SMC 2004), October 10-13, The Hague, The Netherlands.

4. M. Janic, F. Kuipers, X. Zhou and P. Van Mieghem, 2002, "*Implications for QoS provisioning based on traceroute measurements*", Proc. of 3nd International Workshop on Quality of Future Internet Services, QofIS2002 [edited by B. Stiller *et al.* in Springer Verlag LNCS 2511], Zurich, Switzerland, October 16-18, pp. 3-14.

5. P. Van Mieghem and M. Janic, 2002, "*Stability of a Multicast Tree*", Proc. of IEEE INFOCOM02, vol. 2, pp. 1099-1108.

6. M. Janic and Z. Dobrosavljevic, "*Application of neural network as an FM demodulator*", Proc.of 7th Telecommunications Forum TELFOR '99, Belgrade, Yugoslavia (not related to this thesis)