

Thesis presented to fulfill the requirements for the degree of
Doctor in Computer Science of Ecole Polytechnique

Analytical Models and Performance Evaluation in Massive Mobile Ad Hoc Networks

Georgios Rodolakis

Defended on december 7th 2006 before the following committee :

Philippe Jacquet, INRIA (advisor)
Anthony Ephremides, University of Maryland (reviewer)
Olivier Festor, LORIA (reviewer)
Wojciech Szpankowski, Purdue University (reviewer)
Khaldoun Al Agha, Université Paris XI (examiner)
Laurent Viennot, INRIA (examiner)

Abstract

In this thesis, we study different aspects of communication protocols for mobile ad hoc networks. Our goal is to provide analytical models for each of these aspects and to combine the models in order to evaluate the entire system's performance. We consider protocols of all layers starting from medium access control. We begin our study with the IEEE 802.11 protocol and we show via analytical means that the channel access delays follow an asymptotic power law distribution. Based on this result, we discuss a cross-layer protocol with the goal to provide delay based QoS guarantees in multi-hop wireless networks. The next topic involves the scalability of link state routing protocols in massive ad hoc networks. We compare theoretical results on the capacity of wireless networks with the achievable bounds when we take into account the protocol operation and overhead. We adapt the information theoretic bounds to multicast communications and we propose MOST, a protocol for multicast routing which yields optimal performance in the asymptotic case of large ad hoc networks. We then study the behavior of TCP and the impact of the heavy tailed delays observed previously in TCP traffic autocorrelations, again in the context of large networks. In the final part we are concerned with network management and organization, in order to provide user services satisfying QoS constraints. We take here a more general approach regarding the network setting, which consists in placing replicated servers in appropriate locations, based on QoS information obtained from lower layers.

Résumé

Dans cette thèse, nous étudions les différents aspects des protocoles de communication pour les réseaux mobiles ad hoc. Notre but est d'établir des modèles analytiques pour chacun de ces aspects et de combiner les modèles pour évaluer la performance du système en entier. Nous considérons les protocoles de toutes les couches, à partir de la couche de contrôle d'accès au canal. Nous commençons notre étude avec le protocole IEEE 802.11 et nous démontrons que les délais d'accès au canal suivent une distribution polynomiale. Basés sur ce résultat, nous présentons un protocole inter-couche afin d'offrir des garanties de qualité de service de délai dans les réseaux sans fil multi-sauts. Le prochain sujet abordé est la scalabilité des protocoles de routage d'état de liens dans les réseaux ad hoc massifs. Nous comparons les résultats théoriques connus sur la capacité des réseaux sans fil avec les bornes atteignables quand on tient compte du trafic de contrôle des protocoles utilisées. Nous adaptons les bornes théoriques à la communication multicast et nous proposons MOST, un protocole multicast qui atteint des performances asymptotiquement optimales dans les grands réseaux mobiles ad hoc. Ensuite, nous étudions le comportement du protocole TCP et l'impact des délais polynomiaux observés précédemment par rapport aux autocorrélations du trafic TCP, toujours dans le contexte de grands réseaux. Finalement, nous nous intéressons à l'organisation et la gestion du réseau, afin d'offrir des services de qualité garantie. Notre approche peut être appliquée dans un contexte général et consiste à placer des serveurs répliqués dans le réseau, selon les informations de qualité de service fournies par les couches inférieures.

Contents

Preface	1
Thesis Organization and Contributions	2
1 Introduction to Wireless Network Protocols	5
1.1 Network Protocols Architecture	5
1.2 Wireless Network Technologies	7
1.3 Mobile Ad Hoc Networks (MANETs)	8
1.3.1 Routing in Mobile Ad Hoc Networks	9
1.3.2 Optimized Link State Routing (OLSR) Protocol	10
2 Delay Asymptotics and Routing in 802.11 Multi-hop Networks	15
2.1 IEEE 802.11 MAC Protocol	17
2.1.1 Distributed Coordination Function	17
2.1.2 Exponential Back-off Procedure	18
2.2 Asymptotic Delay Analysis	19
2.2.1 One-hop Delay Analysis	19
2.2.2 Multi-hop Delay Analysis	25
2.2.3 Simulation Results	27
2.3 Optimal Delay Based Routing	33
2.3.1 Cross-layer Delay Estimation Protocol	33
2.3.2 Delay Distribution Based Routing	35
2.4 Conclusion	36

3	Scalability Optimizations for Massive Wireless Networks	39
3.1	Neighborhood Management Optimization	40
3.1.1	Modeling Massively Dense Ad Hoc Networks	40
3.1.2	Minimizing the Number of Retransmissions	43
3.2	Scalability of Routing Protocols	45
3.2.1	OLSR Scalability	46
3.2.2	Fish Eye OLSR	49
3.2.3	Useful Capacity	51
3.3	Conclusion	52
	Appendix: Factor λ in $r(\lambda)$	53
4	Multicast Scaling Properties in Large Ad Hoc Networks	55
4.1	Asymptotic Multicast Properties in Ad Hoc Networks	56
4.1.1	Multicast Cost Scaling Law	56
4.1.2	Capacity of Multicast Communication	60
4.1.3	Numerical Results	61
4.2	Specification and Simulation of MOST Protocol	65
4.2.1	Overlay Tree Construction	66
4.2.2	Specification of MOST Protocol	69
4.2.3	Implementation Overview	71
4.2.4	Simulation Results	72
4.3	Conclusion	77
5	Analysis of Traffic Autocorrelations in Large Networks	79
5.1	Autocorrelations Due to MAC Protocols	80
5.1.1	Throughput Analysis	82
5.1.2	Autocorrelation Analysis	86
5.2	Autocorrelations in TCP Traffic	90
5.2.1	TCP Protocol Overview and Models	90
5.2.2	Autocorrelation Function of a Single TCP Connection	95

5.2.3	Long Term Dependencies in Multi-user TCP	98
5.3	Conclusion	100
6	Replicated Server Placement with QoS Constraints	103
6.1	Problem Formulation	105
6.1.1	Optimization Problem Formulation	106
6.1.2	Problem Decomposition	108
6.2	Pseudopolynomial Algorithm	110
6.2.1	Pseudopolynomial Algorithm for Problem 1	110
6.2.2	Pseudopolynomial Algorithm for Problem 2	112
6.2.3	Pseudopolynomial Algorithm for Problem 3	112
6.3	Polynomial Algorithm	115
6.3.1	Polynomial Algorithms for Problems 1 and 2	116
6.3.2	Polynomial Algorithm for Problem 3	117
6.4	Numerical Results	122
6.5	Conclusion	124
	Appendix: Proofs of the Lemmas	125
	Conclusions and Perspectives	135

Preface

The main focus of this thesis consists in the analytical modeling of mobile ad hoc networks. Ad hoc networks are autonomous self-organizing systems of mobile nodes which can communicate without the need of any fixed infrastructure or central control entity. This is made possible by incorporating routing capabilities in the mobile nodes, thus forming multi-hop wireless networks.

With the generalization of the use of mobile devices and the emergence of new wireless Internet services, we expect the importance of wireless networks to continue to grow in the future. In this context, ad hoc networks constitute an interesting solution for providing wireless connectivity to the increasing numbers of mobile devices. However, the deployment of large scale networks presents important challenges, and these challenges are even more intensified in the harsher mobile wireless environment. This accentuates the importance of performance considerations regarding the various protocols at use in this case.

Consequently, in this thesis we study different aspects of communication protocols in the context of very large mobile ad hoc networks. Our goal is to provide analytical models for each of these aspects and to combine the models in order to evaluate the entire system's performance. We take the approach of evaluating the protocols performance in an asymptotic setting, as is the case in the analysis of algorithms. Although, it could be argued that the real life networks are finite in size and therefore such an approach is not realistic, we believe that the opposite is true. As discussed previously, we expect in the future a massive emergence of wireless networks. Moreover, an asymptotic approach makes it possible to identify which system parameters have the most important impact on the network performance in general cases. Hence, we are able to simplify the problems significantly without loss of any meaningful information, and to provide solutions that can scale naturally to large networks, or to identify in which situations the protocol operation may reach a performance limit. In fact, for several problems considered in this thesis, although the theoretical analysis was performed in an asymptotic setting, we were able to verify the results with simulations of realistic situations.

We consider protocols of all layers starting from medium access control. Therefore, the thesis is organized following the layer stack in ascending order. We gradually take a more abstract and general approach with regards to the problem formulations, as we

move up in the layers. However, we comment on the relevance of previously derived results in more specific contexts. Another goal of our work is to utilize the analysis in order to propose practical solutions to those specific situations, in the form of protocols or algorithms.

We begin with an analysis of channel access mechanisms and their interaction with routing protocols, with the goal to provide delay based QoS guarantees in multi-hop wireless networks. The next topic involves the scalability of routing protocols in massive ad hoc networks. We compare theoretical results on the capacity of wireless networks with the achievable bounds when we take into account the protocol operation and overhead. We adapt the information theoretic bounds to multicast communications and we propose a protocol for multicast routing which yields optimal performance in the asymptotic case of large ad hoc networks. We then study the behavior of transport protocols in this context, based on the lower layer protocol models. In the final part we are concerned with network management and organization, in order to provide user services satisfying QoS constraints.

Thesis Organization and Contributions

We present now the contents of each chapter in more details, and we outline our main contributions.

Chapter 1: Introduction to Wireless Network Protocols

In this chapter, we describe the way that protocols are organized in layers and we give a brief overview of the basic operation of Internet protocols that will be studied in the following sections. This discussion permits to place the different thesis contributions in perspective and to outline the organization of this document with regards to the layer stack. We then introduce the concept of mobile ad hoc networks in the context of other wireless technologies. We also give a detailed description of the Optimized Link State Routing (OLSR) protocol, which constitutes the basis of most of the work in this thesis.

Chapter 2: Delay Asymptotics and Routing in 802.11 Multi-hop Networks

This chapter addresses the problem of the evaluation of the delay distribution via analytical means in IEEE 802.11 wireless ad hoc networks. We show that, under certain assumptions, the asymptotic delay distribution can be expressed as a power law. Based on the latter result, we present a cross-layer delay estimation protocol and we derive new delay distribution based routing algorithms, which are well adapted to the QoS requirements of real-time multimedia applications. In fact, multimedia services are not sensitive to average delays, but rather to the asymptotic delay distributions. Indeed, video streaming applications drop frames when they are received beyond a delay

threshold, determined by the buffer size. Although delay distribution based routing is an NP-hard problem, we show that it can be solved in polynomial time when the delay threshold is large, because of the asymptotic power law distribution of the link delays. The theoretical study has been presented in [3]. The cross-layer protocol will be published in [8].

Chapter 3: Scalability Optimizations for Massive Wireless Networks

In this chapter, we study the macroscopic behavior of ad hoc networks in a large scale. We establish a model in which the global network performance can be optimized based on local neighbor tuning adjustments. Moreover, we study the scalability of link state routing and OLSR in particular, and we compare their expected performance with recently obtained theoretical bounds on how well wireless networks could scale. Gupta and Kumar have shown that the total capacity of wireless networks grows according to the square root of the number of nodes. We analyze how this bound is not reached in this case, and we study how much the scalability is enhanced with the use of Fish Eye techniques in addition to the link state routing framework. We show that with this enhancement, the theoretical scalability bounds are reached. In fact, our model permits to quantify the total network capacity in a concise way, *i.e.*, to determine the value of the constant factor in front of the square root capacity formula. The results have been published in [1].

Chapter 4: Multicast Scaling Properties in Large Ad Hoc Networks

In this chapter, we study the benefits of multicast routing in the performance of mobile ad hoc networks, by remaining in the context of massive wireless networks. In particular we show that if a node wishes to communicate with n distinct destinations, multicast can reduce the overall network load by a factor $O(\sqrt{n})$, when used instead of unicast. One of the implications of this scaling property consists in a significant increase of the total capacity of the network for data delivery. We present MOST, a Multicast Overlay Spanning Tree routing protocol based on OLSR, which achieves the previously derived asymptotically optimal capacity results. We perform simulations of the MOST protocol under the ns-2 simulator to verify the theoretical results, and we present a fully working implementation for real network environments. The theoretical study was presented in [4]. The work concerning the protocol will be published in [9].

Chapter 5: Analysis of Traffic Autocorrelations in Large Networks

This chapter addresses the problem of evaluating the performance of the transport layer TCP protocol, and of characterizing analytically the autocorrelation structure of traffic in large networks. At first, we characterize the impact of MAC protocols in both wired and wireless contexts, such as Ethernet and IEEE 802.11, and we show that they can be a

cause for long term dependencies. We then show that, under simple models, a single TCP connection generates traffic with an exponentially decreasing autocorrelation function. However, several TCP connections sharing a given link can generate traffic with long term dependencies, under the condition that the distribution of their round trip delays is heavy tailed. Part of this work was presented in [5].

Chapter 6: Replicated Server Placement with QoS Constraints

In this final chapter, we focus on the efficient provision of services to the network users. Thus, we revisit the issue of delay-based routing in a general setting, not specific to wireless networks. We address the problem of placing replicated servers with QoS constraints. Each server site may consist of multiple server types with varying capacities and each site can be placed in any location among those belonging to a given set. Each client can be served by more than one locations as long as the request round-trip delay satisfies predetermined upper bounds. Our main focus is to minimize the cost of using the servers and utilizing the link bandwidth, while serving requests according to their delay constraint. This is an NP-hard problem. A pseudopolynomial and a polynomial algorithm that provide guaranteed approximation factors with respect to the optimal for the problem at hand are presented. The results were presented in [6] and [2].

Chapter 1

Introduction to Wireless Network Protocols

In recent years, rapid advancements in wireless communication technologies have resulted in a generalization of the use of mobile devices and a vast proliferation of wireless networks. Mobile devices offer possibilities for new applications and it is expected in the future that the driving factor in the Internet's growth will come from wireless technologies, with the emergence of services offered over wireless connections. In this chapter we present an introduction to the organization of the main protocols which are in use in the Internet. Within this context, we present the currently prominent wireless network technologies and standards. We also introduce the concept of mobile ad hoc networks and we give a detailed description of the Optimized Link State Routing Protocol (OLSR), which constitutes the basis of our work in this thesis.

1.1 Network Protocols Architecture

To reduce their design complexity, networks are organized as a stack of *layers*. The purpose of each layer is to offer certain services to the higher layers, keeping the details of the actual implementation hidden. Therefore, a layer on one machine carries on a conversation with a corresponding layer on another machine. The set of rules and conventions used in this conversation are defined by this layer's *protocol*. In Figure 1.1 we present a model of the Internet protocols' layered architecture [99], as well as the classification of some protocols that we study in this thesis.

In the following we describe the functions of each layer and we outline the operation of some protocols that will be studied in further detail in other chapters. We also comment on the thesis organization, which in fact follows the protocol stack.

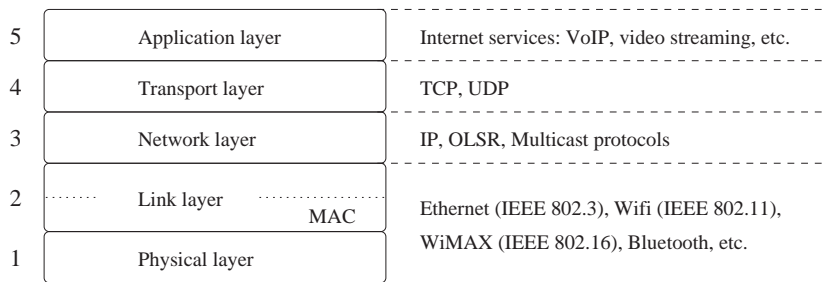


Figure 1.1: Layers and protocols considered in the context of this work.

Layer 1: The Physical Layer This layer is concerned with encoding and transmitting data over a communication channel. The physical layer protocols are in charge of moving the raw bits representing the data from one device to another.

Layer 2: The Link Layer The link layer protocols are responsible for forming and monitoring frames (ordered sequences of bits) transferred over a link connecting the sender and the receiver. Broadcast networks have an additional issue: how to control access to the shared channel. A special sublayer of the link layer, the Medium Access Control (MAC) sublayer, deals with this problem. An example of a layer 2 protocol for wired networks is Ethernet (*cf.* Section 5.1). As shown in Figure 1.1, the wireless protocols that we study in this work are situated in this layer too. In the next section we will describe the different types of wireless technologies and the corresponding protocol standards. In fact, the standards generally refer to physical as well as MAC/link layer specifications. However, in this thesis we are mostly interested in the MAC protocols. IEEE 802.11 is a layer 2 protocol that we will study extensively in Chapter 2.

Layer 3: The Network Layer This layer connects machines using possibly different technologies, creating the abstraction of a network. A key issue is determining how packets are routed from source to destination. The most important layer 3 protocol is the Internet Protocol (IP). IP defines an official packet format and identifies each machine with a unique identifier: the IP address. This addressing makes routing possible. Therefore, the OLSR routing protocol which we present in this chapter is a network layer protocol. IP also defines identifiers for groups of machines, in order to allow multicast, *i.e.*, one to many communication. Thus, we place multicast routing protocols in the network layer too. Due to the importance of routing, layer 3 considerations are present throughout the thesis when we study other layers. Conversely, this layer is our main focus in Chapters 3 and 4.

Layer 4: The Transport Layer At a basic level, this layer accepts incoming data from applications, fragments it into discrete packets and passes each one on to the

network layer; at the other end it reassembles the received packets into the output stream. In fact, the transport layer is a true end-to-end layer, all the way from the source to the destination. In other words, a program on the source machine carries on a conversation with a similar program on the destination machine. Hence, the transport layer is responsible for distributing the right packets to to the right application. This distinction is done by adding to packets application-specific identifiers, called ports. The two main layer 4 protocols in the Internet are UDP and TCP. UDP (User Datagram Protocol) is an unreliable, connectionless protocol. TCP (Transmission Control Protocol) offers more transport layer functionalities. It is a reliable connection-oriented protocol that allows a byte stream originating on one machine to be delivered without error on any other machine in the Internet. TCP also handles flow control to make sure a fast sender cannot swamp a slow receiver with more messages than it can handle. The TCP protocol will be described in more detail and analyzed in Chapter 5.

Layer 5: The Application Layer This layer contains the protocols that directly provide services to the users. One widely-used application protocol is HTTP (HyperText Transfer Protocol), which is the basis for Web browsing. Other examples of services are VoIP (Voice Over IP), video streaming *etc.* We will be concerned with how to manage and organize such services in Chapter 6.

1.2 Wireless Network Technologies

Wireless technologies can be placed at the bottom of the layer stack. In the Internet context, they define the physical and medium access functions, and can be used in conjunction with the TCP/IP protocols. We discuss here some of the current wireless technologies and standards which have been developed for several practical purposes, ranging from short-ranged device interconnection to wide area voice and data networks.

Wireless Local Area Networks (WLANs) are the wireless equivalent of local area networks, which are usually privately-owned networks within a single building or campus, of up to a few kilometers in size. The prominent IEEE 802.11 standard [14] has become synonymous with WLANs due to its wide-spread success over the last years. It is more commonly known as *Wifi* and constitutes today a widely deployed solution for providing wireless access to notebook computers in offices, public spaces, university campuses *etc.* The name IEEE 802.11 corresponds to the working group in charge of the standardization in the IEEE (Institute of Electrical and Electronics Engineers). The standardization process began in the mid nineties, resulting in a first version of the 802.11 protocol in 1997 running at either 1 Mbps or 2 Mbps. From 1999 to 2003 the standards committee proposed three high-speed amendments: 802.11a operating in the wider 5 GHz ISM frequency band and delivering 54 Mbps, 802.11b in the 2.4 GHz ISM band achieving up to 11 Mbps, and 802.11g which shares the same frequency spectrum as 802.11b and the same physical layer technology as 802.11a for rates up to 54 Mbps.

The proposed standards include physical layer and MAC layer specifications. In this thesis, we are interested in the MAC layer of the protocol. The 802.11 MAC protocol can operate in two modes. In the *base station mode*, all communications go through the base station, called an access point in 802.11 terminology. In the *ad hoc mode*, the computers communicate with each other directly. The MAC specification defines two channel access methods: the Point Coordination Function (PCF) where transmissions are coordinated by the access points, and the Distributed Coordination Function (DCF) which is based on the CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) mechanism. However, PCF mode is optional and it is rarely implemented. A detailed description of the DCF mode will be presented in Chapter 2.

Another kind of wireless networks is utilized for system interconnection without wires, within an operating space of a few meters: for example, a wireless network connecting a computer with its mouse, keyboard, and printer. A technology which permits to create small range networks for wireless device interconnection, with rates that can reach 1Mbps, is Bluetooth [11].

A category of wireless networks spanning over large geographical areas are the cellular telephone networks. In addition to these low-speed networks, high-bandwidth metropolitan area networks are also being developed, with the goal to provide the users with wireless broadband Internet connectivity. A technology being developed for this purpose is the IEEE 802.16 standard [15][16], also known as WiMAX which operates in the 10 to 66 GHz frequency range, in areas of several square kilometers.

A characteristic of the described current technologies is the need for a fixed infrastructure. Base stations or access points synchronize the wireless devices and allow them to form a network. However, in some situations this approach is not practical and there is a need to deploy self-organizing wireless multi-hop networks. For example, small inexpensive electronic devices, like fire or intrusion detection sensors, can be distributed over large areas and monitor different events. In order to collect the data in an efficient and cost-effective way, the devices can be organized in *sensor networks* and relay the information over multi-hop wireless paths to a data collection center. For the more complex situation where the devices are mobile and with heavier data transport requirements, we describe in the next section Mobile Ad hoc Networks (MANETs) and we mention some potential applications.

1.3 Mobile Ad Hoc Networks (MANETs)

A Mobile Ad hoc Network (MANET) is an autonomous system of mobile nodes which can move freely and communicate over wireless links, without any fixed infrastructure or central control [41]. In contrast to the wireless networks already described, MANETs must support robust and efficient operation in a mobile wireless environment by incorporating routing functionality into the nodes. Such networks are envisioned to have dynamic, sometimes rapidly-changing, random, multi-hop topologies which are

likely composed of relatively bandwidth-constrained wireless links.

Mobile ad hoc networks are suited for situations where a fixed infrastructure is not available, not trusted, too expensive or unreliable. A MANET can operate either in isolation, or as a “stub” network connecting to a fixed internetwork. Some examples of applications include: military or emergency networks, the interconnection of notebook computers or PDAs in a conference or campus, highway networks, inexpensive alternatives or enhancements to cell-based mobile network infrastructures, as well as extensions to WLANs when radio coverage is difficult, *etc.*

The challenges of ad hoc networking are different from those of wired networks and hence most Internet solutions that were developed and applied successfully on wired networks fail in the harsher mobile ad hoc environment. We mention some salient MANET characteristics, as described in [41]:

- Dynamic topologies,
- Bandwidth-constrained, variable capacity links,
- Energy-constrained operation,
- Limited physical security.

Moreover, since some envisioned MANETs may be relatively large, another requirement for these networks is the ability to scale, which is a characteristic that will be addressed in Chapter 3 of this thesis.

1.3.1 Routing in Mobile Ad Hoc Networks

As a result of the previously described characteristics and the promising application prospects, ad hoc networking has attracted significant research interest and routing protocols have been developed specifically for MANETS. The protocols can be classified essentially in two main categories: proactive protocols and reactive protocols. Each approach has its own advantages and their relative performance depends on the network topology, mobility of the nodes, as well as the traffic patterns.

Proactive protocols maintain a routing table with paths to every destination in the network. For this purpose the protocol needs to continuously exchange information between nodes about the network topology. The downside of this approach is the overhead which is generated by the protocol. However MANET protocols optimize the bandwidth utilization. On the other hand, there is the advantage that routes are available instantly when a they are needed. An example of a proactive routing protocol is OLSR (Optimized Link State Routing) [37].

Reactive protocols operate on-demand, *i.e.*, routes are calculated according to user demands and the routing tables which are maintained are incomplete. This reduces

the protocol overhead in case only a few routes in the network are in use, at the cost of additional delay. An example of a routing protocol using the reactive approach is AODV (Ad hoc On-Demand Distance Vector Routing) [91].

In the following section, we give a description of the OLSR proactive routing protocol, since the presented protocol solutions to different ad hoc network issues that we study (for instance QoS, multicast, or scalability) are proposed as extensions to OLSR.

1.3.2 Optimized Link State Routing (OLSR) Protocol

The Optimized Link State Routing Protocol (OLSR) [37] operates as a table driven, proactive protocol, *i.e.*, exchanges topology information with other nodes of the network regularly. This protocol is actually an optimized version of the link state routing protocol for wired networks [86]. This optimization is achieved with the utilization of the concept of *multipoint relays* (MPR). Each node selects a set of its neighbor nodes as MPRs. In OLSR, only nodes, selected as such MPRs, are responsible for forwarding control traffic, intended for diffusion into the entire network. MPRs provide an efficient mechanism for flooding control traffic by reducing the number of transmissions required. For its core operation, OLSR uses two types of control messages: Hello messages and TC messages.

Unified Packet Format

OLSR communicates using a unified packet format for all data related to the protocol. The purpose of this is to facilitate extensibility of the protocol without breaking backwards compatibility. This also provides an easy way of piggybacking different types of information into a single transmission, and thus for a given implementation to optimize towards utilizing the maximal frame-size, provided by the network. These packets are embedded in UDP datagrams for transmission over the network. Each packet encapsulates one or more messages. The messages share a common header format, which enables nodes to correctly accept and (if applicable) retransmit messages of an unknown type. In Figure 1.2 we depict the basic layout of any packet in OLSR (omitting IP and UDP headers).

The **Message Type** field indicates which type of message is to be found in the MESSAGE part. The **VTime** field indicates for how long time after reception a node must consider the information contained in the message as valid. The **Originator Address** contains the main address of the node, which has originally generated this message.

MPR Optimization

MPR nodes are elected by their neighbors because they cover their two-hop neighborhood. Therefore, MPR selection is performed in an entirely distributed manner. The

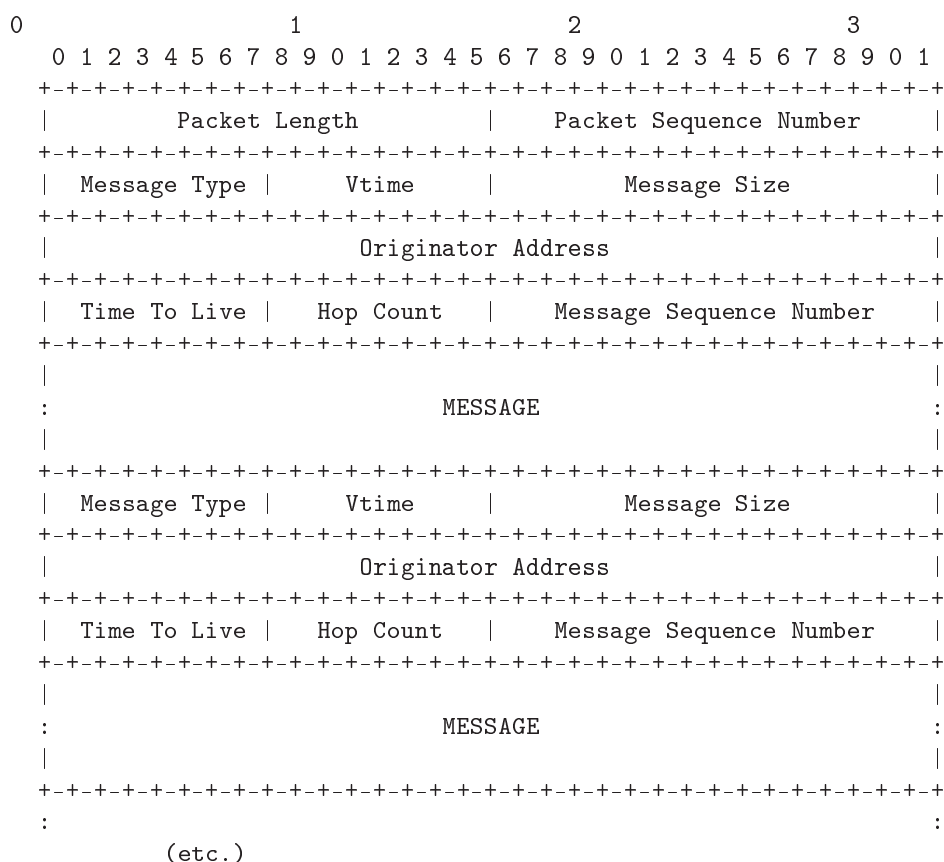


Figure 1.2: Generic OLSR packet format.

protocol uses the MPRs to facilitate efficient flooding of control messages in the entire network. Moreover, in route calculation, the MPRs are used to form the route from a given node to any destination in the network.

A node selects MPRs from among its one hop neighbors with *symmetric*, *i.e.*, bi-directional, linkages. Therefore, selecting the route through MPRs automatically avoids the problems associated with data packet transfer over uni-directional links (such as the problem of not getting link-layer acknowledgments for data packets at each hop, for link-layers employing this technique for unicast traffic).

Once the MPR set is determined, optimized flooding can be achieved if each node retransmits a broadcast packet whenever it receives its first copy from a neighbor that has selected the node as a multipoint relay. In Figure 1.3, we depict an example of the gain that can be achieved by using MPR flooding instead of a pure flooding mechanism.

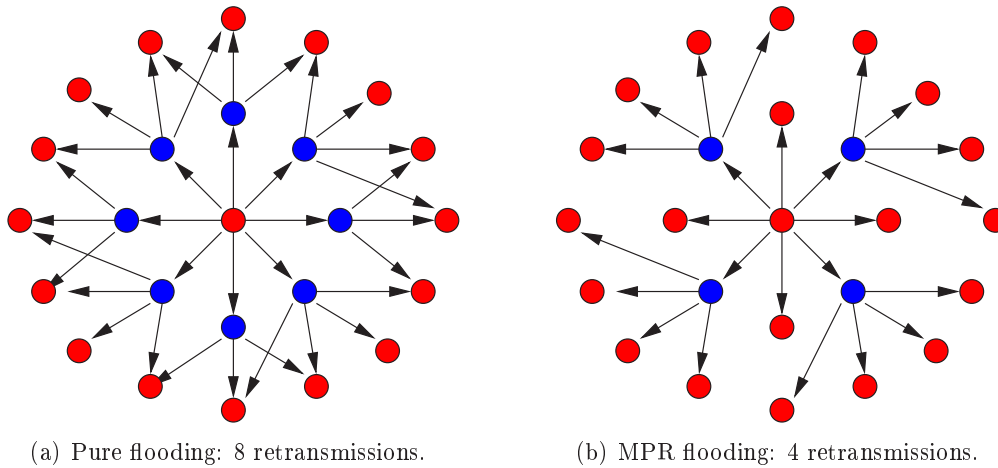


Figure 1.3: Diffusion of a message using pure flooding and MPR flooding.

Neighbor Sensing

Hello messages have the role of neighbor detection, as well as determining the status of the links. They are exchanged periodically between neighboring nodes, without being forwarded. In Hello messages, nodes advertize their neighborhood information. Therefore, using Hello messages, a node can determine which nodes are its symmetric neighbors and select its MPRs. The MPR information is also advertized in the Hellos, hence nodes can determine which neighbors have selected them as MPRs.

To ensure that the quality of the links established by OLSR is good, a *link hysteresis* strategy is proposed in advanced neighbor sensing. Essentially, an additional parameter is associated with each link, which defines the link quality. This parameter's estimation can be done using the percentage of Hello messages that are actually received from each neighbor. If this value exceeds a certain threshold (HYST_THRESHOLD_HIGH) the link can be established. On the other hand, when the quality drops below the minimum threshold (HYST_THRESHOLD_LOW) the link is considered to be in a pending state and it is not used for data forwarding. Additionally, a similar strategy can be implemented using the signal to noise ratio.

Topology Dissemination

Each node maintains topological information about the network, obtained by means of TC (Topology Control) messages. TC messages are flooded in the network periodically taking advantage of MPRs to reduce the control traffic overhead. Furthermore, only MPR nodes have the responsibility to declare link state information in the network. In order to save more on control traffic, nodes have the possibility to advertize a small

subset of their neighbor links. The advertized link set can be limited to MPR links, *i.e.*, the neighbors that have elected this node as an MPR. In this case the nodes have only a partial knowledge of the network topology. The fact that any given node can compute a shortest path to any arbitrary destination comes from the fact that the node knows its own neighbor list. However there is an option to advertize additional information, as the whole neighbor list.

Chapter 2

Delay Asymptotics and Routing in 802.11 Multi-hop Networks

In this chapter, we begin our study on the performance of wireless networks. We model the performance of layer 2 and 3 protocols and their impact on the delay in the delivery of data packets. This delay is of primary importance for a wide range of applications, such as VoIP, interactive multimedia, video streaming, *etc.* In Table 2.1 we give an example of delay requirements for packet delivery in voice applications. Similar requirements can be formulated for other parameters such as the jitter, *i.e.*, the variance of the delay. We note that this example may be used as an indication for other interactive applications as well, since the threshold after which the delay starts becoming noticeable by the user is 100 ms. Therefore, the delay in the delivery of packets must not exceed a certain threshold, which depends on the application, in order to achieve acceptable quality of service.

Table 2.1: Delay requirements for voice traffic.

Delay	Perceived quality
< 100-150 ms	Delay not detectable
150-250 ms	Delay noticeable, quality still acceptable
> 250-300 ms	Unacceptable delay

With the wide deployment of wireless networks in home and office networks, as well as the expected growth of mesh and ad hoc networks, the possibility of supporting real-time applications becomes a crucial requirement in this context. Hence, we investigate the performance of wireless networks utilizing the IEEE 802.11 protocol (most commonly known as *Wifi*). Our goal is to analyze the impact of the random channel access mechanism on the packet delays, and to use the analysis to provide optimal routing solutions for multi-hop wireless networks, since in this case a detailed understanding of

16 Delay Asymptotics and Routing in 802.11 Multi-hop Networks

the impact of both the underlying channel access and routing protocols on the delay characteristics is essential. More precisely, we address the problem of the analytical evaluation of the delay distribution in a multi-hop wireless network with IEEE 802.11 MAC protocol [14] under the Optimized Link State Routing (OLSR) protocol. The main channel access mechanism in IEEE 802.11 is the Distributed Coordination Function (DCF), which is a Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) scheme. DCF was designed initially for asynchronous traffic and since it is a random access protocol it does not provide any guarantee for delay sensitive applications.

Most of previous research about the performance of the 802.11 protocol concerns single hop networks. A simple analytical model of the 802.11 DCF access mechanism was introduced in [28] and was used to analyze the saturation throughput performance. The MAC layer service time was studied in [107], by expanding the previous model. The delay in both saturated and unsaturated networks was also studied in [103], where each node was modelled as a discrete time queue. In the case of multi-hop networks, the majority of QoS solutions proposed are concerned with bandwidth requirements (see [88] for a study in the 802.11 context). A protocol which supports multiple metric routing criteria (including the average delay) is the QOLSR extension [26, 25] to OLSR. Our goal in this thesis is to propose a delay based routing solution specific to IEEE 802.11 networks, in order to achieve the best possible performance.

Therefore, we evaluate the performance of the 802.11 protocol in the context of wireless ad hoc networks, and we obtain both one-hop and multi-hop delay estimates. We denote by W the end-to-end delivery delay of a packet. We analyze the delay distribution $P(W > T)$ and we show that in the case that T is large (*i.e.*, several times the average delay) the probability $P(W > T)$ decays as a power law, namely in T^{-a} , where a is a constant. Based on the analysis, we present a cross-layer framework, which takes into consideration all aspects of the network protocols in use, to evaluate the delay distribution $P(W > T)$ for any large T and we use it in order to find routes that satisfy given delay requirements. A delay-oriented quality of service for a connection is generally expressed via a maximum acceptable delay T and a maximum over-delay ratio ϵ , specified by the application, requiring that during the connection the constraint $P(W > T) < \epsilon$ is verified. In general finding the optimal route that minimizes an over-delay ratio is NP-hard [57]. Nevertheless, the fact that the delay distribution at every node router is in power law allows us to specify a polynomial approximation algorithm with an approximation factor of $1 + O(T^{-1})$.

The chapter is organized as follows. In Section 2.1, we present an overview of the IEEE 802.11 MAC protocol. In Section 2.2 we introduce the general model framework and we analyze the one-hop as well as the multi-hop delay distribution. The analysis is verified via simulations. In Section 2.3 we describe a cross-layer delay estimation protocol based on OLSR, and we indicate how the previous analysis can be used in delay distribution based routing.

2.1 IEEE 802.11 MAC Protocol

A general introduction of the IEEE 802.11 protocols was presented in Chapter 1. Since the context of this chapter is the study of ad hoc networks, we focus on the DCF mode, of which we give a more detailed description.

2.1.1 Distributed Coordination Function

The Distributed Coordination Function (DCF) is the fundamental access method used in the IEEE 802.11 MAC protocol. It is based on the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) mechanism, which is designed to reduce the collisions due to multiple sources transmitting simultaneously on a shared channel. In the CSMA/CA protocol, a station transmits only if the medium is idle. The medium is considered as idle if it is sensed to be idle for a duration greater than the Distributed Inter-frame Space (DIFS). If the medium is sensed as busy, the transmission is deferred until the end of the ongoing transmission. When the medium becomes idle after a busy period, the node does not transmit immediately, because multiple stations could have been waiting for the end of the transmission and may attempt to access the channel again at the same time. Therefore, the node starts a random wait by initializing its *back-off timer*. The back-off timer is randomly selected in an interval called the *contention window* and has the granularity of one *slot*. Every time the channel is sensed to be idle, the back-off counter is decremented. When the counter reaches zero, the node can start its transmission. If the channel is sensed as busy during the back-off procedure, the counter is frozen and then resumed when the channel becomes idle for a DIFS again. To make sure that a transmitted *unicast* frame has reached its destination, an *Acknowledgement* frame is generated from the destination to the source. This frame is sent after a time interval equal to a SIFS (Short InterFrame Space), which is shorter than a DIFS, effectively giving higher priority to Acknowledgement frames. The source node can detect lost frames if after a fixed time interval, called *AckTimeout*, an acknowledgement has not been received. On the other hand, *broadcast* frames are not acknowledged, hence they are not retransmitted in case there is a collision. The channel access mechanism is summarized in Figure 2.1, where we depict the medium occupancy during a successful unicast frame transmission (*i.e.*, where no collisions occur).

The above carrier sense is called physical carrier sense because it is performed at the air interface. A virtual carrier sense is also possible in the DCF mode to resolve the problem of the hidden terminal. This problem occurs when two nodes that are not within hearing distance of each other create collisions at a third terminal that receives the transmission from both. The virtual carrier sense is performed at the MAC sublayer. The channel is reserved before each transmission, so instead of transmitting the data frame after sensing that the channel is idle, the station sends an RTS (Request To Send) frame to the destination. The receiver replies by a CTS (Clear To Send) frame, after which the data transfer can start. However, the use of RTS/CTS frames imposes additional delay

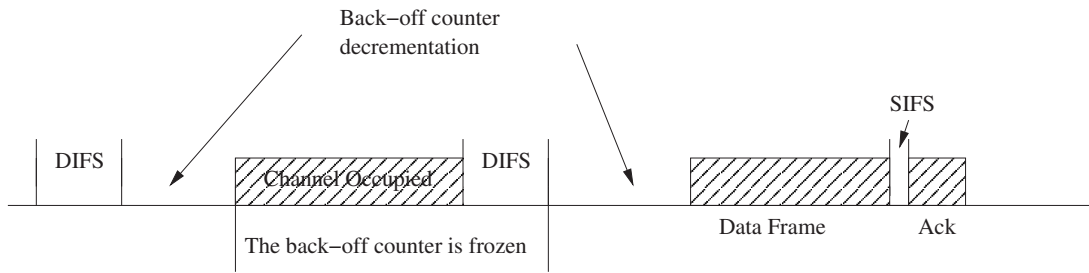


Figure 2.1: Example of medium occupancy in a successful unicast transmission using the DCF channel access mechanism.

and bandwidth overhead. Therefore the RTS/CTS mechanism is recommended only for big packets.

2.1.2 Exponential Back-off Procedure

As discussed previously, the wireless nodes maintain a rotating back-off timer in order to randomize their access to the channel and reduce the collision probability. According to the CSMA protocol, the back-off timer is selected in an interval $\{0, \dots, CW - 1\}$, where CW is the contention window. In spite of that, collisions can still occur. In order to reduce the probability of further collisions, the contention window is doubled after each collision to increase the random waiting time.

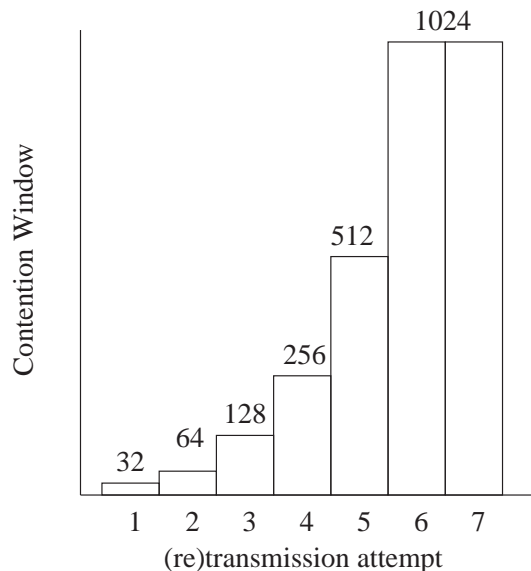


Figure 2.2: Binary exponential back-off.

Initially the contention window is set to CW_{\min} . If a collision occurs the nodes select a new back-off number in an enlarged interval $\{0, \dots, 2CW_{\min} - 1\}$. The contention window length is multiplied by two after each collision until it reaches the maximum value CW_{\max} . When the frame is transmitted successfully or the maximal number of retransmissions is reached, the back-off interval length is reset to CW_{\min} for the next packet. The retransmission limit is defined in the standard depending on the size of packets. For packets longer than the RTS threshold, *i.e.*, packets which are preceded by an RTS/CTS exchange, the limit is 4, while for shorter packets the limit is 7. The exponential back-off procedure is illustrated in Figure 2.2, where according to the specification $CW_{\min} = 32$ and $CW_{\max} = 1024$.

2.2 Asymptotic Delay Analysis

In this section, we carry out the analytical study of end-to-end delays in 802.11 multi-hop networks. In order to simplify the formula derivations, we perform the analysis under certain modeling assumptions. For instance, we use an M/G/1 queueing model for the nodes, which cannot be considered *a priori* entirely realistic. However, we have verified via simulations that our model is pertinent and can accurately predict the shape of the node delay distributions in the domain of interest. Furthermore, the assumptions we make are not fundamental for our results, therefore we comment on plausible model generalizations whenever possible. We focus on the asymptotic case of large delays, which permits to single out which system parameters have the most significant influence on the delay performance.

2.2.1 One-hop Delay Analysis

Methodology Overview

A wireless node can be seen as a buffer filled by incoming messages and with a single server that performs the CSMA/CA multiple access protocol. We model this system as an M/G/1 queue, *i.e.*, we assume:

1. The input packet flow in the buffer is Poisson of rate λ packets per slot;
2. Service delays are independent.

In fact, the M/G/1 hypothesis is just a matter of simplifying approach. Since we are going to deal with heavy tailed distribution of service times, the consequence on queueing time distribution can be generalized to a much larger class of queueing models. For example it is not necessary to assume independence between service times or to restrict to Poisson input in order to derive a power law queueing distribution (but in this case the coefficients change). Nonetheless, as we verify later in the simulations section, the M/G/1 hypothesis leads to satisfactory results.

Service Delay Determination

The IEEE 802.11 CSMA/CA protocol uses a rotating back-off mechanism where the nodes have to wait a random number of idle slots between transmission attempts. Let \mathbf{C} be the random variable that expresses the number of busy slots between two consecutive idle slots. Let $p(L)$ be the probability of collision that is experienced by packets, when the packet length is L (that is the transfer time of the packet in the channel, expressed in slots, which is proportional to its size in bits). The longer the packet is, the more likely it is to collide. We take the following assumptions:

1. Durations between successive idle slots are independent and i.i.d;
2. Collision events on successive transmissions of the same packet are independent.

In practice there is a maximum number of retransmissions after which the packet is discarded in case of permanent failure. The default maximum retry is 7 and can lead to considerable delays. Since this delay is larger than the maximum acceptable delay we think of regarding connection QoS, it does not practically matter to set the maximum number of retries to infinity.

Let $C(z)$ be the probability generating function $\sum_n P(\mathbf{C} = n)z^{n+1}$, quantity C being expressed in slot duration. This generating function corresponds to the time needed for a back-off counter decrease, expressed by the random variable $C + 1$ (we add one slot to the quantity C for the decrease to be taken into account). Identity $C(z) = z$ would mean that $\mathbf{C} = 0$ always, *i.e.*, the channel is permanently sensed idle (note that in this case one slot is still needed for the counter decrease).

Let $\beta(z, L, p, k)$ be the probability generating function of the service delay when the packet length is L , the collision probability is p and the initial back-off interval is k . The service delay of a packet corresponds to the time elapsed since it was extracted from the buffer until it is transmitted successfully. Therefore, it takes into account retransmissions due to collisions and it includes the time needed to access to the channel (corresponding to the rotating back-off decrementation) plus the fixed packet transmission length.

We will express all these quantities using generating functions, starting from the time needed to access the channel, or equivalently the back-off counter decrease. As discussed earlier, each back-off decrease is expressed by the random variable $C + 1$, with generating function $C(z)$. If the back-off counter is i , the total time to access the channel is the time needed for i counter decreases, or the sum of i times the random variable C . From the independence assumption it comes that in this case the channel access time can be expressed by generating function $C(z)^i$. Since the initial back-off window is k , and the back-off counter value is selected uniformly at random in the interval $\{1, \dots, k\}$ (we also take here into account the DIFS interval), the generating function of the total channel access time can be written as $\frac{1}{k} \sum_{i=1..k} C(z)^i$, which results from the previous discussion by taking either possible value for i with probability $\frac{1}{k}$. Once the channel is

accessed the time needed to transmit the packet is fixed and equal to L^1 , therefore it can be expressed by generating function z^L . Hence the the service time when no collision occurs comes from adding the previous two quantities, or equivalently the corresponding generating function is equal to the product of the above generating functions, *i.e.*,

$$\frac{z^L}{k} \sum_{i=1..k} C(z)^i = \frac{C(z)^{k+1} - C(z)}{C(z) - 1} \frac{z^L}{k}. \quad (2.1)$$

In order to account for packet collisions, we obtain the following recursion:

$$\beta(z, L, p, k) = \frac{C(z)^{k+1} - C(z)}{C(z) - 1} \frac{z^L}{k} \times (1 - p + p\beta(z, L, p, 2k)). \quad (2.2)$$

In case there is no collision (with probability $1 - p$), the service delay corresponds to our previous calculations. The term $\beta(z, L, p, 2k)$ is obtained from the case where there is a collision (with probability p), hence the procedure is repeated after doubling the back-off interval and this results in an additional service delay term.

The service delay probability generating function is

$$\beta(z) = E[\beta(z, L, p(L), CW_{\min})],$$

which is obtained by averaging on packet length L and collision probabilities $p(L)$.

Figure 2.3 shows the 200 first coefficients of $\beta(z)$ when $C(z) = 0.8z + 0.2z^4$, $L = 4$ and $p = 0.3$. In this theoretical example, the packet transferring time is 4 slots, and each decrementation of the back-off counter takes one slot with probability 80% and 4 slots with probability 20% (which means that the channel is busy with a packet transmission). The coefficients in this figure were obtained from numerical calculations using Maple, by iterating the recursive equation (2.2).

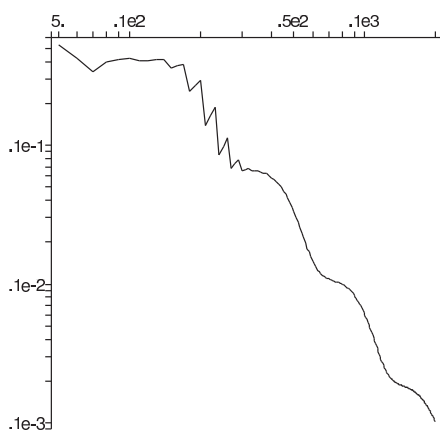


Figure 2.3: Coefficients of $\beta(z)$.

¹ L can be adjusted to include *AckTimeouts* too.

Delays Including Queueing

In order to compute the delay experienced by packets in the buffer, we take the formula for slotted M/G/1 for the queue delay probability generating function $q(z)$:

$$q(z) = \exp\left((\beta(z) - 1)\frac{\lambda}{2}\right) \frac{(1 - \lambda\beta'(1))(1 - z)}{1 - z \exp(-(\beta(z) - 1)\lambda)}. \quad (2.3)$$

This needs the provision that $\beta'(1)$ exists. We will see that this implies that $p < \frac{1}{2}$. Similarly, for the existence of the k^{th} moment of service time we need that $p < 2^{-k}$. If $\lambda \ll 1$ then we can replace (2.3) by:

$$q(z) \approx \frac{(1 - \lambda\beta'(1))}{1 - \frac{z}{1-z}(1 - \beta(z))\lambda}. \quad (2.4)$$

The generating function of the overall delay, so called one-hop node delay (queueing + service), of a packet of length L with collision probability p , $w(z, L, p)$ satisfies the identity:

$$w(z, L, p) = q(z)\beta(z, L, p, CW_{\min}). \quad (2.5)$$

Figure 2.4 shows the coefficients of $w(z)$ for $\lambda = 0.02$. Notice that $\beta'(1) = 22.939\dots$.

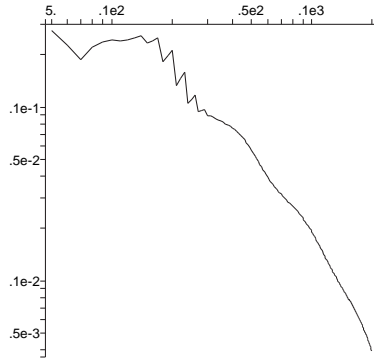


Figure 2.4: Coefficients of $w(z)$.

Asymptotic Analysis

We denote S the service time and W the overall delay in a router. In this section we derive asymptotic estimates for the distributions of the above quantities by applying Flajolet-Odlyzko theorems [50].

Theorem 1 *We have the expansion for z around 1:*

$$\beta(z, L, p, k) = 1 + (1-z)v(1-z) + (kC'(1)(1-z))^B \alpha(\log(1-z)) + O((1-z)^{B+1}),$$

where $v(x)$ is a polynomial, $B = -\log_2 p$ assuming that B is not integer, and $\alpha(x)$ is a periodic function of period $\log 2$ with small fluctuation.

Proof: We fix L and p and set $e^{-\theta} = C(z)$ and denote $j(\theta, k) = \beta(z, k)$. We have

$$j(\theta, k) = \frac{1 - e^{-k\theta}}{k\theta} f(\theta)(1 - p + pj(\theta, 2k)),$$

with $f(\theta) = e^\theta \frac{\theta}{1 - e^{-\theta}} z^L$.

It is clear that

$$\theta = (1-z)C'(1) + O((1-z)^2).$$

We define $g(\theta) = \prod_{i \geq 1} \frac{1 - e^{-\theta 2^{-i}}}{\theta 2^{-i}}$. Thus if $\nu(\theta, k) = g(k\theta)j(\theta, k)$, then

$$\nu(\theta, k) = g(2k\theta)f(\theta)(1-p) + pf(\theta)\nu(\theta, 2k).$$

And $\nu(\theta, k) = \frac{1-p}{p} \sum_{i \geq 1} (f(\theta)p)^i g(2^i k\theta)$.

It can be proven that function $g(\theta)$ is analytical and behaves like $1 + O(\theta)$ when $\theta \rightarrow 0$ and converges to zero faster than any power law when $\theta \rightarrow \infty$.

Let $r_B(\theta)$ be the polynomial of degree $\lfloor B \rfloor$, which is the Taylor expansion of $g(\theta)e^\theta$ at $\theta = 0$. Recall that $B = -\log_2 p$.

Let $g_B(\theta) = g(\theta) - r_B(\theta)e^{-\theta}$. Clearly $g_B(\theta) = O(\theta^{\lfloor B \rfloor})$ when $\theta \rightarrow 0$.

We have

$$\nu(\theta, k) = u_B(\theta) + \frac{1-p}{p} \sum_{i \geq 1} (f(\theta)p)^i g_B(2^i k\theta),$$

with

$$u_B(\theta) = \frac{1-p}{p} \sum_{i \geq 1} (f(\theta)p)^i r_B(2^i k\theta) e^{-2^i k\theta}.$$

Clearly $u_B(\theta)$ is an analytical function with $u_B(\theta) = 1 + O(\theta)$.

Let

$$\nu_B(\theta, k) = \frac{1-p}{p} \sum_{i \geq 1} (f(\theta)p)^i g_B(2^i k\theta).$$

We will show that $\mu_B(\theta, k) = \theta^{-B} \nu_B(\theta, k)$ is bounded when $\theta \rightarrow 0$. Let $h_B(\theta) = \theta^{-B} g_B(\theta)$.

24 Delay Asymptotics and Routing in 802.11 Multi-hop Networks

We have

$$\mu_B(\theta, k) = \frac{1-p}{p} \sum_{i \geq 1} (f(\theta))^i h_B(2^i k \theta) k^B.$$

Since $f(\theta) = 1 + O(\theta)$, when $\theta \rightarrow 0$, we have $\mu_B(\theta, k)$ which converges to $\alpha(\log \theta) = \sum_i h_B(2^i k \theta) k^B$, the sum being on all integers i , including the negative integers. The sum converges because $h_B(\theta) = O(\theta^\epsilon)$ with $\epsilon = \lceil B \rceil - B$ and $h_B(\theta)$ decays faster than any power law. Notice that function $\alpha(x)$ is periodic of period $\log 2$.

Therefore $j(\theta, k) = \frac{\nu(\theta, k)}{g(k\theta)}$ has asymptotic expansion

$$\frac{u_B(\theta)}{g(k\theta)} + \alpha(\log \theta) \theta^B + O(\theta^{B+\epsilon}).$$

The theorem follows with a change of variable.

Theorem 2 *The probability that the service time is greater than T , for T large is*

$$P(S > T) = (CW_{\min} C'(1))^B \alpha^*(\log T) T^{-B} + O(T^{-B-1}),$$

where $\alpha^*(x)$ is also a periodic function of period $\log 2$ with small fluctuation.

Proof: From the previous theorem it comes that $\beta(z)$ fits Flajolet-Odlyzko asymptotic conditions [50].

By writing the function $\alpha(\log(1-z))$ as a Fourier series:

$$\alpha(\log(1-z)) = \sum_n \alpha_n (1-z)^{\frac{2in\pi}{\log 2}},$$

and applying Flajolet-Odlyzko theorems, we have

$$P(S > T) = (W_{\min} C'(1))^B \alpha^*(\log T) T^{-B} + O(T^{-B-1}),$$

where α^* is periodic in $\log T$, of period $\log 2$:

$$\alpha^*(\log(T)) = \sum_n \frac{\alpha_n}{\Gamma(2-B-\frac{2in\pi}{\log 2})} T^{\frac{2in\pi}{\log 2}}.$$

Theorem 3 *We have the expansion for z around 1:*

$$\begin{aligned} w(z) &= 1 + (1-z)u(1-z) \\ &\quad + \frac{\lambda(CW_{\min} C'(1))^B}{1-\lambda\beta'(1)} \alpha(\log(1-z))(1-z)^{B-1} \\ &\quad + O((1-z)^B), \end{aligned}$$

where $u(x)$ is an analytic function.

Proof: We substitute the expansion for $\beta(z)$ around $z = 1$, derived in Theorem 1, in the formula for $q(z)$ given by (2.4). The theorem follows by using the expansion around $z = 1$ in equation $w(z) = q(z)\beta(z)$.

Theorem 4 *The probability that the delay in a router is greater than T , for T large is*

$$P(W > T) = \frac{\lambda(CW_{\min}C'(1))^B}{1 - \lambda\beta'(1)}\alpha^*(\log T)T^{1-B} + O(T^{-B}).$$

Proof: We use the result of the previous theorem and we apply Flajolet-Odlyzko theorems on $w(z)$.

Notice that the delay distribution tail decays in power law. As a corollary it turns out that the existence of the k th moment of the delay needs $p < 2^{-k-1}$. Also, to obtain the asymptotic delay distribution estimate, only the average of the channel occupancy distribution $C'(1)$ is required, rather than the distribution $C(z)$.

Remark The latter assume non integer B , otherwise we have:

$$\begin{aligned} w(z) &= 1 + (1-z)u(1-z) \\ &\quad + \frac{\lambda(CW_{\min}C'(1))^B}{1 - \lambda\beta'(1)}\alpha_1(\log(1-z))\log(1-z)(1-z)^{B-1} \\ &\quad + O((1-z)^B). \end{aligned}$$

and in this case:

$$P(W > T) = \frac{\lambda(CW_{\min}C'(1))^B}{1 - \lambda\beta'(1)}\alpha_1^*(\log T)T^{1-B} + O(T^{-B}).$$

In the following we assume that B is non-integer.

2.2.2 Multi-hop Delay Analysis

We now compute the end-to-end delay distribution for any given route in the network, based on the one-hop delay analysis discussed previously. For this purpose, we assume that the M/G/1 model remains valid for each node. This is the strongest hypothesis in our analysis since the traffic coming from relay nodes is not Poisson. The Markovian input can be justified theoretically as the sum of a large number of independent random traffics generated by the neighboring nodes. In practice, even if this is not completely verified, the model still gives a very satisfactory approximation, as discussed in Section 2.2.1. Furthermore, we assume that when travelling on its route, the delay

26 Delay Asymptotics and Routing in 802.11 Multi-hop Networks

experienced by a packet on a router is independent of the delay experienced on the other routers. This latter assumption makes the problem easier to handle mathematically. However it is not fundamental for our result, since it is known that the sum of two random variables in power law is still in power law whatever the dependence assumptions between them. The power law in the resulting distribution function will be the maximum of the respective power laws of the variables, except that the factor in front of it will depend on the dependence assumptions. To see this, consider two random variables X_1, X_2 , such that $P(X_1 > T) \propto T^{-B_1}$ and $P(X_2 > T) \propto T^{-B_2}$. We will show that $X_1 + X_2$ is also in power law.

We have the lower power law bound: $P(X_1 + X_2 > T) \geq P(X_1 > T)$. Also, we can obtain an upper power law bound since:

1. $X_1 + X_2 \leq 2 \max\{X_1, X_2\}$,
2. $P(\max\{X_1, X_2\} > T) = P(X_1 > T \wedge X_2 > T) \leq P(X_1 > T) + P(X_2 > T)$.

Assuming independence from now on for simplification purposes, if there are n routers in the route from the source to the destination then the probability generating function of the end-to-end delay is equal to the product $\prod_{i \in \text{route}} w_i(z)$, where $w_i(z)$ is the probability generating function of the delay at router number i and *route* is a set of router indices.

Still, with Flajolet Odlyzko result [50], if each $w_i(z)$ is of the form :

$$1 + (z - 1)g_i(z) + c_i(z - 1)^{B_i - 1} + O((z - 1)^{B_i}),$$

then the leading term of $P(W(\text{route}) > T)$ is

$$\sum_{i \in \text{route}} c_i^* T^{1 - B_i},$$

with $c_i^* = \frac{c_i}{\Gamma(2 - B_i)}$.

Keeping only leading terms:

$$P(W(\text{route}) > T) \approx c(\text{route})T^{1 - B(\text{route})},$$

where $B(\text{route}) = \min B_i$ and $c(\text{route}) = \sum_{B_j = B} c_j^*$.

An unexpected consequence of the above is that a good choice for the route should not be the shortest path in number of hops. In the shortest path the lap between two consecutive routers may be too large, leading to too large collision rates and therefore a too low value of $B(\text{route})$. If we take shorter hops between routers, then we will reduce the collision rate and get a larger value of $B(\text{route})$. Of course this would be done in the detriment of a larger number of hops and a larger value of $c(\text{route})$. But, since in $c(\text{route})T^{1 - B(\text{route})}$ parameter T is supposed to be large, the reduction of $T^{1 - B(\text{route})}$

would prevail in most cases on the $c(\text{route})$ increase. Interestingly enough, increasing the number of hops and $c(\text{route})$ will in most cases increase the *average* end-to-end delay. Therefore we have the paradoxical case where increasing the average delay actually decreases the over-delay loss ratio. This is due to the fact that we expect the average delay to be much lower than the maximum acceptable delay T . Consequently, routing with respect to the average delay may conflict with the minimization of the over-delay ratio.

Conversely, the optimal route may be too long since it may have too short hops. In this case the connection may waste too many resources. Instead of choosing the route that minimizes $P(W > T)$ it is probably wiser to seek the shortest route that satisfies the requirement $P(W > T) \leq \epsilon$.

2.2.3 Simulation Results

We use the ns-2 [46] simulator to validate our delay modeling. We study various scenarios for different purposes. We compare the analytic service time distribution with the measured service time distribution (obtained by ns-2 simulations). We aim to show that the service time and the sojourn time (in other words the delay including queuing too) are in power law. Furthermore, we investigate whether one-hop delays are independent within a route, and finally we show that the end-to-end delay is in power law too. Common simulation parameters are summarized in Table 2.2.

Table 2.2: Simulation Settings.

MAC Parameters	Wmin 32, slot 20μ s
Propagation model	Two ray ground
Transmission range	250m
Packet size	1000 bytes
Traffic type	Exponential (Poisson)
Simulation time	300s
Simulation area	$800 \times 800 m^2$
Routing protocol	OLSR

One-hop Delay Measures

In the first scenario, we consider an ad hoc network with 5 nodes as shown in Figure 2.5. The 802.11 bandwidth is 1Mb. Five exponential flows with 140kbs data rate are launched between different pairs of nodes (represented by arrows in Figure 2.5). In order to study the cumulative delay distribution in node 2, we measure the main parameters in this node for the conducted simulation, as presented in Table 2.3.

28 Delay Asymptotics and Routing in 802.11 Multi-hop Networks

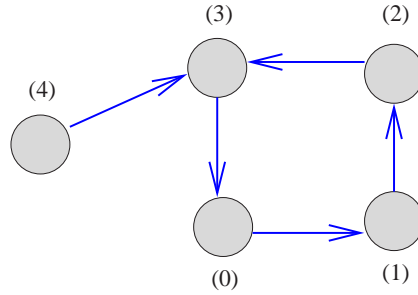


Figure 2.5: Topology 1.

Table 2.3: Measured parameters.

Channel occupancy	$C(z) \simeq 0.82z + 0.04z^{16} + 0.03z^{125} + 0.1z^{445}$
Packets per slot	$\lambda = 0.00024$ packets/slot (12 packets/second)
Collision probability	$p = 0.09, B = 3.45$

Based on these parameters and (2.2), we use Maple to compute the service time distribution, which we draw in Figure 2.6. The service time distribution measured via ns-2 simulation is shown in Figure 2.7. To demonstrate that the service time distribution is in power law with $B = -\log_2(p)$ (here $B = 3.45$), as stated in Theorem 2, we draw the equation $Y = \alpha X^{-3.45}$, where α is a constant, and we compare the two plots. Figure 2.7 shows that, for T large enough, the service time distribution and Y have the same power law exponent.

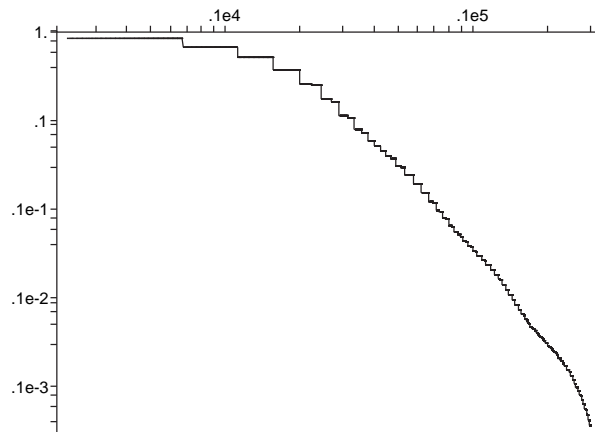


Figure 2.6: Analytic service time distribution.

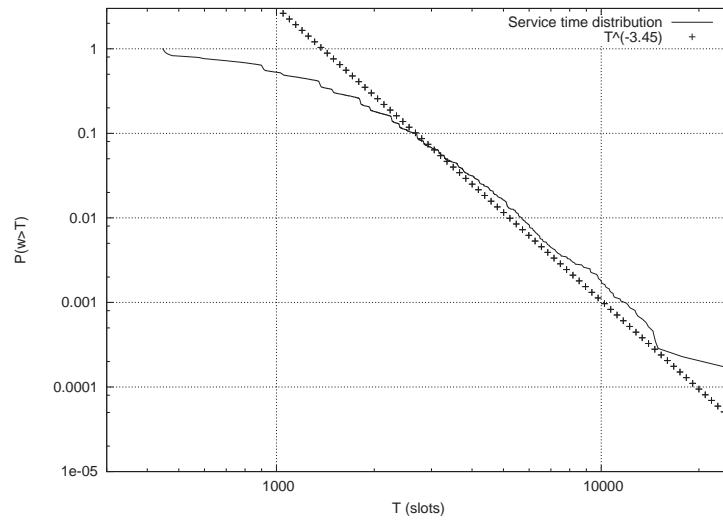


Figure 2.7: Measured service time distribution.

In the same way, we measure the sojourn time distribution (also called node delay) which we present in Figure 2.8. We notice that for T between 4000 and 40000 slots (*i.e.*, 80ms to 800ms), the node delay is in power law with exponent $1 - B = -2.45$, in accordance with Theorem 4.

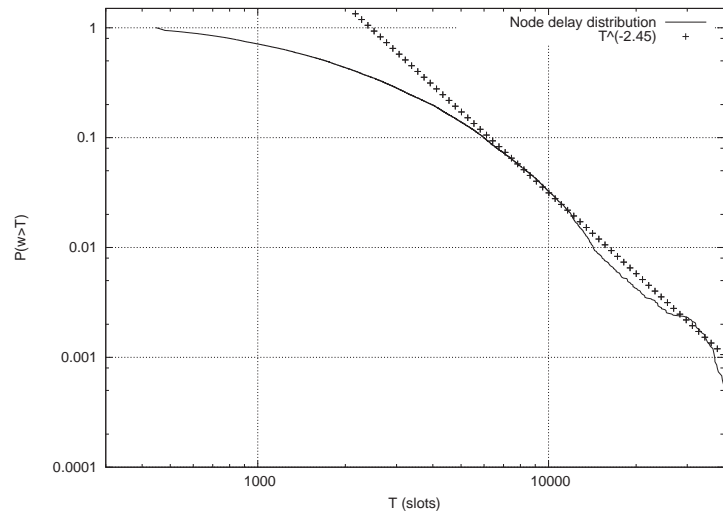


Figure 2.8: Measured node delay distribution.

Multi-hop Delay Measures

Secondly, we consider a randomly generated topology of 50 nodes which is depicted in Figure 2.9. We launch 10 exponential flows in the network and we measure for each the end-to-end delay distribution. We run several simulations by varying the throughput from 2 to 8 packets per second. We consider the flow following the five hop path shown in Figure 2.9. We measure the end-to-end delay of this flow as well as collision probabilities along the path (for each hop). Table 2.4 summarizes the obtained probabilities.

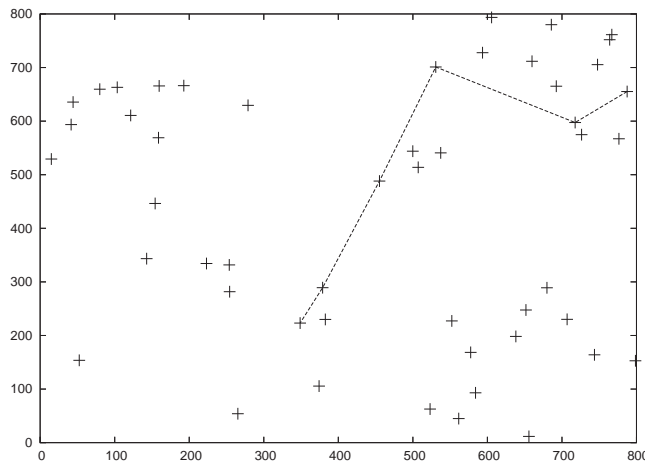


Figure 2.9: Topology 2.

Table 2.4: Collision probabilities for each hop of the path.

Throughput	Per hop collision probabilities %				
2pkt/s	0.53	0.94	0.19	1.05	1.13
8pkt/s	1.11	2.28	0.45	5.43	5.75

Figure 2.10 compares the measured end-to-end delay distribution with theoretical results for sending rates of 2 and 8 packets per second respectively. According to the analysis of multi-hop delay distribution in Section 2.2.2, the power law exponent is equal to $1 + \log_2(p)$ such that p corresponds to the highest collision probability along the path. Referring to Table 2.4, the highest collision probability on this path is 0.0113 ($1 - B = -5.46$) and 0.0575 ($1 - B = -3.12$) for traffic rates of 2 and 8 pkt/s respectively.

We also measure the single hop delay distributions along the path connecting the source to the destination. Let $W_i, i = 1 \dots 5$, be the distribution generating functions for each hop and W the end-to-end delay distribution generating function. We compute the

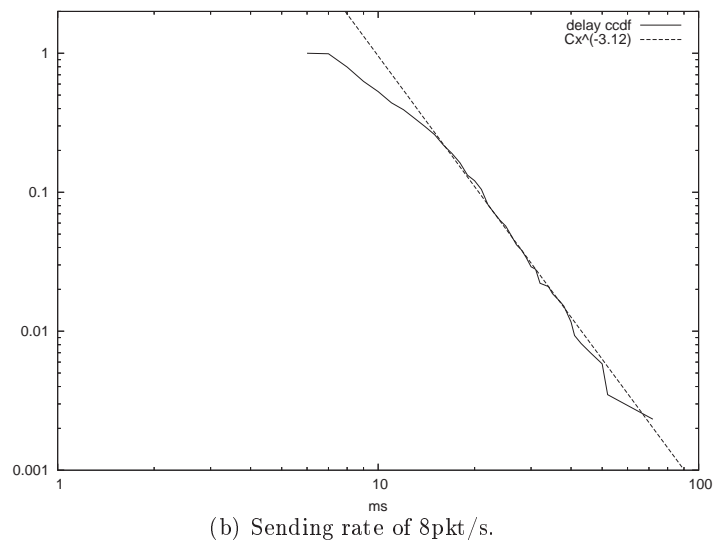
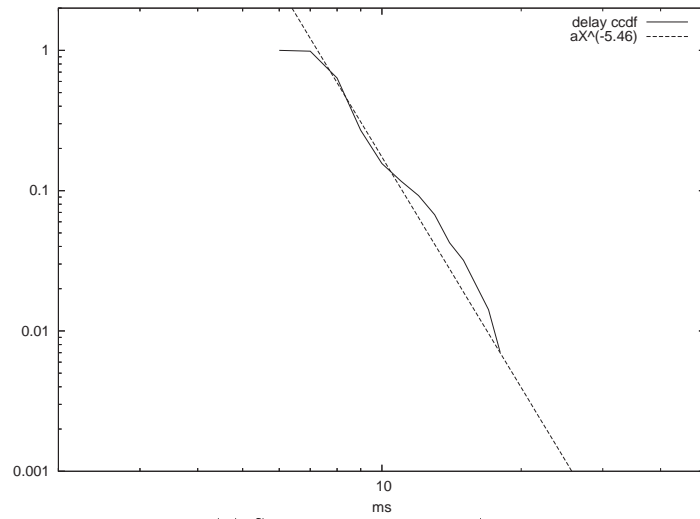
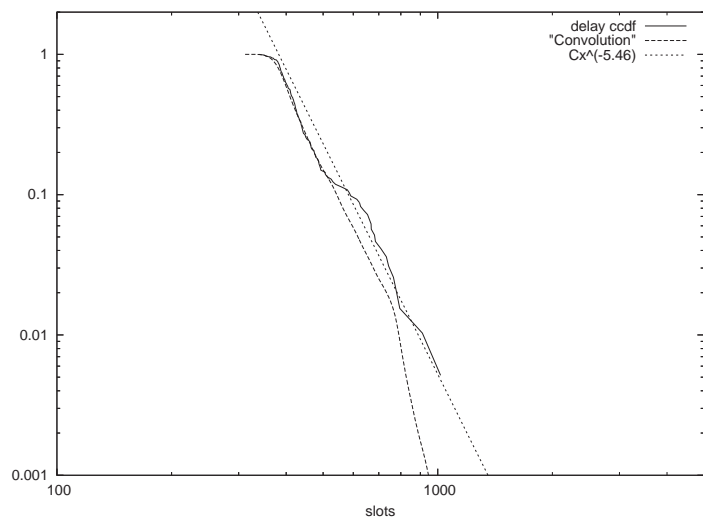
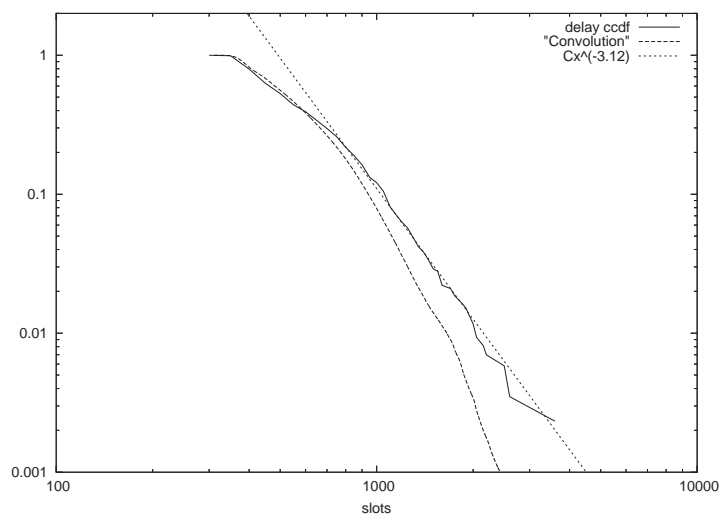


Figure 2.10: Comparison between end-to-end delay distribution and the corresponding power law.

32 Delay Asymptotics and Routing in 802.11 Multi-hop Networks



(a) Sending rate of 2pkt/s.



(b) Sending rate of 8pkt/s.

Figure 2.11: Comparison between the end-to-end distribution and the convolution of single hop distributions.

product $\prod_{i=1..5} W_i$ and we compare it to W . In case the delays are independent within a route, the above product corresponds to the end to end delay distribution, since it expresses the convolution of random variables W_i , $i = 1..5$. As shown in Figure 2.11, the curves representing W and $\prod_{i=1..5} W_i$ are slightly different which means that there is a weak dependence between single hop delays, yet it is weaker when the network is lightly loaded. Notice that even when the independence assumption is not completely verified, the delay is still a power law as explained in Section 2.2.2.

2.3 Optimal Delay Based Routing

In this section, we present a delay estimation protocol, used to obtain estimates of the delay distributions for all routes in the network in a proactive way, and subsequently we show how this information can be used to optimize route computation. In fact, we propose an extension to the OLSR routing protocol to support delay estimation for any given route.

2.3.1 Cross-layer Delay Estimation Protocol

As mentioned previously, the single hop delay distribution estimate is based on the knowledge of the collision probability and the average of the channel occupancy distribution C , which are basically MAC layer parameters. The multi-hop delay distribution is based on the knowledge of single hop characteristics along a given route, clearly concerning the functioning of the routing protocol. Therefore, the extended protocol needs to interact with the MAC layer. In Figure 2.12, we depict the protocol framework.

Average of Channel Occupancy Distribution C

The channel occupancy information concerns the internal functioning of wireless cards and is not actually known. However, the card acknowledges successful frame transmissions by sending special interrupts to the driver. This allows to measure the service time of transmitted packets. Knowing the service time, it is possible to deduce the time needed to access the channel in case of broadcast packets such as OLSR Hello messages (since they are not retransmitted when a collision occurs). Thus, based on (2.1), it is possible to derive the mean of the channel occupancy distribution C from the mean of the Hello access time distribution, noted by μ_{hello} . We have $C'(1) = \frac{2\mu_{hello}}{CW_{min}+1}$.

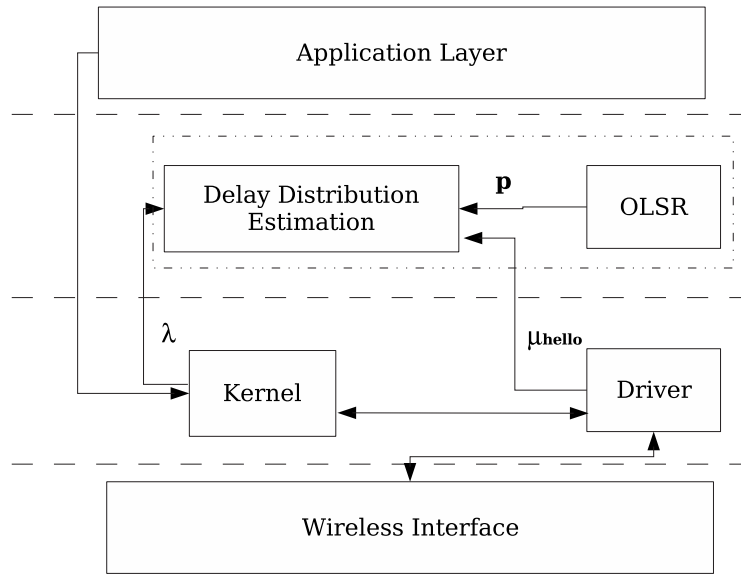


Figure 2.12: Delay estimation protocol framework.

Collision Probability Estimation

The collision probability is estimated by OLSR, since this information is not currently provided by wireless cards. OLSR has a procedure in the advanced neighbor sensing option that allows to compute the collision rate of Hello messages (link quality level parameter, *cf.* Section 1.3.2). It uses the Hello message sequence number in order to identify the missing Hellos. However there could be a difficulty in the fact that the collision probability $p(L)$ may depend strongly on packet length L . One may expect a dependence of the kind $-\log p(L) = aL + b$ where a and b are scalar coefficients. Since the neighbor has no idea of the size of missing Hellos, the transmitter should advertize the length distribution of its Hellos. Comparing with its received Hello distribution the neighbor would be able to determine the coefficients a and b . By default the neighbor assumes $a = 0$, *i.e.*, all packets have the same collision rate regardless of their length.

Advertizing Link Quality

Multi-hop delay computation is based on the knowledge of the one-hop delays of the route. Thus, each node must inform the entire network of its local information. Therefore, we introduce a Link Quality Advertizement (LQA) message, which is broadcasted in the network via the optimized MPR-flooding mechanism of OLSR. For our purpose, it is preferable to use the option full-OLSR, *i.e.*, to advertize the whole neighbor set instead of the MPR selector set. The node advertizes for each link ℓ the collision rate p_ℓ , and for itself it advertizes the global λ , provided by the kernel as shown

in Figure 2.12, and the value of $C'(1)$. Another alternative is to advertize directly the tuple $(p, \frac{\lambda(W_{\min}C'(1))^B}{1-\lambda\beta'(1)})$. Using the information collected from LQA messages, each node can calculate the delay distribution for any route in the network.

2.3.2 Delay Distribution Based Routing

In this section, we utilize the knowledge provided by the protocol presented in Section 2.3.1 to perform routing according to delay requirements specified by the application layer. These requirements are expressed as a maximum delay threshold T , and a maximum acceptable ratio of packets ϵ to exceed this threshold. To satisfy this constraint, we need to know the delay distribution in every node. The problem of delay distribution based routing consists in finding a route that satisfies the application end-to-end delay requirement $P(W > T) < \epsilon$ for a given connection. Multiple routes satisfying such a delay constraint can be found in the network, hence a routing algorithm must select one among them. In this section, we explore two possible directions. The first direction consists in finding the optimal route that minimizes $P(W > T)$. The second direction consists in finding the shortest route (in hops) that satisfies the requirement $P(W > T) \leq \epsilon$.

Finding the Optimal Route

In general, finding the optimal route with respect to a delay distribution is NP hard [57]. But if we stick to the asymptotic expression, we can find a polynomial Dijkstra like algorithm. The problem is to find the route that provides the best asymptotic expansion of the quantity $P(W(\text{route}) > T)$ when $T \rightarrow \infty$. By best asymptotic expansion we mean the one that provides asymptotically the lowest $P(W(\text{route}) > T)$. Since we expect that $P(W(\text{route}) > T)$ is asymptotically equivalent to $\sum_{i \in \text{route}} c_i^* T^{1-B_i}$ (cf. Section 2.2.2), the idea consists in minimizing the sum of the leading terms of the one-hop delay distributions along the route. Hence, the routing algorithm is effectively a Dijkstra algorithm, where the weights on the links are $c^* T^{1-B}$. Parameters c^* and B are calculated according to the analysis in Section 2.2.2. The weight of the route is the sum of the weights of the links, and the optimal route is the route that minimizes this sum. When T is finite, the sum of the weights on a route gives an approximation of the end-to-end delay distribution within a factor $1 + O(T^{-B(\text{route})})$, according to the asymptotic analysis. Since $B(\text{route}) > 1$, the algorithm is optimal within a factor $1 + O(T^{-1})$.

Finding the Shortest Route Satisfying the Delay Constraint

As discussed previously, the shortest route that satisfies the constraint $P(W > T) \leq \epsilon$ is generally preferable to the much longer route that minimizes the quantity $P(W > T)$. Moreover, a major problem due to the use of any dynamic metric is route fluctuation.

36 Delay Asymptotics and Routing in 802.11 Multi-hop Networks

However, the proposed routing on the shortest path that verifies an over-delay ratio constraint provides more stable routes. In the previous section we described a polynomial search algorithm which is optimal within a factor $1 + O(T^{-1})$, hence for T sufficiently large the search provides the optimal route. In the present section we aim to find the shortest route according to a certain additive metric on links, *i.e.*, the number of hops, which satisfies a given constraint according to another additive metric, *i.e.*, the quantities c^*T^{1-B} . In general such a multi-metric optimization problem is again NP-hard. However since the first metric can only take integer values we can easily make it polynomial using dynamic programming.

We model the network as a weighted graph. We consider a source node s . We denote by $v_j, j = 1..n$, all the nodes in the network, where v_1 is the source s . Each link connecting two nodes (v_i, v_j) is associated to a weight w_{v_i, v_j} which corresponds to the asymptotic probability c^*T^{1-B} .

For each node v_j we define $p(i, v_j)$ as the smallest known value according to the sum of the weights w , of all routes of length i that connect the source node s to node v_j . Note that $i \leq n$, since the longest path in the network is at most n hops.

We describe the algorithm as follows. We initialize all values $p(0, v_j)$, for $j = 1..n$ to infinity except for $p(0, v_1) = 0$. We will compute $p(i, v_j)$ for all i, j . For each $i \geq 1$ in increasing order, we compute the values $p(i, v_j), j = 1..n$, using equation

$$p(i, v_j) = \min_{c \in \mathcal{N}(v_j)} (p(i-1, c) + w_{c, v_j}),$$

where $\mathcal{N}(v_j)$ is the neighborhood of node v_j and w_{c, v_j} is the weight of the link (c, v_j) . Notice that for all values of i that are smaller than the distance between s and v_j we have $p(i, v_j) = \infty$.

The aggregate computational cost of $p(i, v_j)$ for all nodes v_j and for a given i is $O(m)$, where m is the total number of links in the network. Hence, in the worst case, the total time needed to construct the table $p(i, v_j)$ for all possible values $i, j = 0..n$ is $O(mn)$. Once the table has been constructed, the shortest route to any destination d satisfying the required delay constraint, corresponds to the route of minimum i such that $p(i, d) < \epsilon$. In case the algorithm computes the route for one particular destination, the iteration on the route length i can stop as soon as a feasible route is found.

2.4 Conclusion

We analyzed the delay distribution in 802.11 multi-hop networks and we demonstrated that for large values of T the cumulative delay distribution $P(W > T)$ is a power law. In practice, simulations show that this is true from T equal to approximately twice the average. The delay distribution for a specific route can be derived based on MAC layer as well as network layer parameters, hence we present a cross-layer solution for a delay estimation protocol as an extension to the OLSR routing protocol. Furthermore, the

information from this protocol can be used to compute the route that satisfies the QoS delay requirements specified by a multimedia application. In fact delay distribution based routing is known to be an NP-hard problem. However, the asymptotic analysis in power law makes it possible to obtain a polynomial, Dijkstra-like, algorithm.

It is important to note that the routing algorithm does not guarantee that the calculated route will satisfy the delay constraint after launching the new traffic. In case the new connection has a significant impact on the network conditions, it is necessary to dynamically control the delay, in order to check whether the constraint is still verified. Due to its proactive nature, the proposed delay estimation protocol allows to compute periodically the end-to-end delay, thus the routes can be readjusted. Therefore, it would be interesting to combine the delay routing protocol with a mechanism providing dynamic delay control, as well as admission control when the connection delay requirements cannot be satisfied.

38 Delay Asymptotics and Routing in 802.11 Multi-hop Networks

Chapter 3

Scalability Optimizations for Massive Wireless Networks

After the detailed protocol analysis in the previous chapter, we now examine the macroscopic behavior of ad hoc networks in a massive scale. Our interest lies in the scalability properties of wireless ad hoc networks. Namely, we study the impact of the large number of nodes in the system performance and the theoretical feasibility of such networks using existing network layer protocols. The most important result about the scalability of wireless networks was obtained by Gupta and Kumar in their seminal paper [58]. They showed via information theory that when the number of nodes N in the network increases (with randomly placed nodes and uniform traffic density), the maximum network capacity per node is $O(\frac{1}{\sqrt{N \log N}})$. This is in contrast to the case where all nodes share a common medium (for example when all wireless nodes are within hearing range), which results in an average $O(\frac{1}{N})$ share of the available capacity. In fact, the capacity of the network increases when the radio ranges decrease due to spatial reuse, in spite of interferences and multihopping. However, there is a lower limit to the radio range since the network must stay connected, which leads to the optimal neighborhood size estimate of $O(\log N)$. This in turn leads to a radio range in $O(\sqrt{\frac{\log N}{N}})$, hence a network diameter in hop number being $O(\sqrt{\frac{N}{\log N}})$, and to the estimation of the maximum per node capacity. If we just require the existence of a giant component instead of a connected network (or if we assume that nodes are optimally placed), the $\log N$ factor can be dropped in the formulas. Indeed, the condition to have a giant component is that the average neighborhood size is greater than 2, and we can therefore consider in our study that it is no longer $O(\log N)$, but rather $O(1)$. Similarly, the Gupta and Kumar scaling property in presence of variable traffic density implies that the optimal neighbor size in presence of traffic λ bits per square area unit per time unit is $O(\frac{1}{\lambda})$.

Therefore, neighborhood management in wireless networks has a significant impact

on network performance. For instance, one of the key issues of ad hoc networking is the lack of bandwidth, which implies the need to reduce the number of packet retransmissions when routing towards a destination. In the first part of this chapter, we examine how to tune neighbor management to optimize this relaying. We discuss how a local neighborhood optimization in a shortest path routing protocol like OLSR can lead to global network performance which is asymptotically optimal, in the absence of QoS mechanisms such as the one introduced in Chapter 2. However, the overhead generated by the neighbor management protocol will have an impact on the neighborhood itself. In the second part of this chapter, we aim to show how a link state routing protocol can fulfill the Gupta and Kumar scaling property, in the context of slotted time ad hoc networks.

3.1 Neighborhood Management Optimization

In this section we describe how we model the different aspects of ad hoc networks, and we discuss how neighborhood management can be optimized with OLSR.

3.1.1 Modeling Massively Dense Ad Hoc Networks

We consider the following model, which was first introduced in [63]: time is slotted and the mobile nodes are all synchronized, *i.e.*, transmissions occur at the beginning of slots and according to an ALOHA-like protocol (*i.e.*, nodes select at random their transmission slots). We consider an area of arbitrary size \mathcal{A} (we will ignore border effect). N transmitters are uniformly distributed. We call λ the density of transmitters per slot and per area unit, and f the rate of packet transmissions per slot and per node. In this model we will assume that the distribution of active transmitters per slot and area unit is a Poisson process.

In order to justify this assumption, note that we have a uniform distribution of nodes and that nodes use an ALOHA-like multiple access scheme. Therefore the number and positions of transmitters at beginning of slots vary with time and changes from slot to slot like a random process. The resulting distribution of transmitters should therefore be exactly identified as a Bernoulli distribution over a uniform distribution. However, this kind of distributions are known to quickly converge to a Poisson distribution when $N \rightarrow \infty$ and $fN/\mathcal{A} \rightarrow \lambda$. Thus we decided to directly work with this approximation, which turns out to be very accurate in practice.

Let X be a node at a random position. We will again ignore border effects and assume that all nodes transmit at the same nominal power. The reception signal at distance r is then $P(r) = r^{-\alpha}$ with $\alpha > 2$. Typically α ranges from 2.5 to 4. Notice that the expression of quantity $P(r)$ does not involve any fading factor. Fading is an alteration of the signal which is due to factors other than the distance (obstacles, co-interferences with echoes, and so on). Fading is generally modeled via the introduction of a non-zero

factor that varies randomly with time and node location. We will also show how to address the fading issue more thoroughly in the following.

Let W be the signal intensity received by node X at a random slot. The quantity W is then a random variable since the number and location of transmitters is random and varies with the slot. Let $w(x)$ be its density function. If we consider \mathcal{A} to be infinite, we can use [63] to compute the Laplace transformation of $w(x)$. We have $\tilde{w}(\theta) = \int w(x)e^{-x\theta}dx$ satisfying the following identity, which takes into account the independent Poisson contributions of all nodes in the infinite area \mathcal{A} :

$$\begin{aligned}\tilde{w}(\theta) &= \exp\left(\int_0^{2\pi} \int_0^\infty \lambda(e^{-\theta r^{-\alpha}} - 1)rdrd\phi\right) \\ &= \exp(2\pi\lambda \int_0^\infty (e^{-\theta r^{-\alpha}} - 1)rdr) .\end{aligned}$$

Then, using standard algebra we get (with a change of variable $u = \theta r^{-\alpha}$ in the above integral):

$$\tilde{w}(\theta) = \exp(-\lambda\pi\Gamma(1 - \frac{2}{\alpha})\theta^{2/\alpha}) . \quad (3.1)$$

Note that if instead of an area, the node location map was a line (for instance a sequence of mobiles nodes on a road) we would then have:

$$\tilde{w}(\theta) = \exp(-\lambda\Gamma(1 - \frac{1}{\alpha})\theta^{1/\alpha}) . \quad (3.2)$$

And similarly, if the location map was a volume (for instance a network formed by aircrafts), we would instead have:

$$\tilde{w}(\theta) = \exp(-\frac{4}{3}\lambda\pi\Gamma(1 - \frac{3}{\alpha})\theta^{3/\alpha}) . \quad (3.3)$$

In the following, we will restrict ourselves to the case where nodes are located on a 2D map.

Neighbor Model

A node is considered to be a neighbor with another node if the probability of successfully receiving hellos from each other is greater than a certain threshold p_0 . For example we can take $p_0 = 1/3$. This can be achieved by keeping track of the hello receival success rate per neighbor, as it is done in the ‘‘advanced neighbor sensing’’ of OLSR, and by appropriately tuning the hysteresis threshold (*cf.* Section 1.3.2).

We will assume that a packet can be successfully decoded if its signal-over-noise ratio is greater than a given threshold K . Typically $K = 10$. Therefore a node will correctly receive a packet from another node at distance r with probability $P(W < r^{-\alpha}/K)$. Since hello packets are never retransmitted, the hello success rate from a node at distance r is

exactly $P(W < r^{-\alpha}/K)$. Therefore nodes at distance r are neighbors as long as $P(W < r^{-\alpha}/K) > p_0$. This is equivalent to $r < r(\lambda)$, where $r(\lambda)$ is the critical radius such that $\int_0^{r(\lambda)^{-\alpha}/K} w(x)dx = p_0$. In fact quantity λ is a parameter which is easy to handle since by simple algebra it comes that $r(\lambda) = \lambda^{-1/2}r(1)$ (see this chapter's appendix). The surface covered by the radius $r(\lambda)$ is then the neighborhood area $\sigma(\lambda) = \frac{\sigma(1)}{\lambda}$.

We will now compute $\sigma(1)$. We remind that factor λ is now omitted ($\lambda = 1$). For simplification purposes, we set $C = \pi\Gamma(1 - \frac{2}{\alpha})$ and $\gamma = \frac{2}{\alpha}$. By application of the reverse Laplace transformation we get:

$$P(W < x) = \frac{1}{2i\pi} \int_{-i\infty}^{+i\infty} \frac{\tilde{w}(\theta)}{\theta} e^{\theta x} d\theta. \tag{3.4}$$

Expanding $\tilde{w}(\theta) = \sum_n \frac{(-C)^n}{n!} \theta^{n\gamma}$, it comes:

$$P(W < x) = \frac{1}{2i\pi} \sum_n \frac{(-C)^n}{n!} \int_{-i\infty}^{+i\infty} \theta^{n\gamma-1} e^{\theta x} d\theta. \tag{3.5}$$

Then by bending the integration path towards the negative axis we get:

$$\begin{aligned} \frac{1}{2i\pi} \int_{-i\infty}^{+i\infty} \theta^{n\gamma-1} e^{\theta x} d\theta &= \frac{\sin(\pi n\gamma)}{\pi} \int_0^\infty \theta^{n\gamma-1} e^{-\theta x} d\theta \\ &= \frac{\sin(\pi n\gamma)}{\pi} \Gamma(n\gamma) x^{-n\gamma}. \end{aligned}$$

Figure 3.1 shows the plot of $P(W < x)$ versus x for $\alpha = 2.5$ and $\lambda = 1$. Let x_0 denote the value such that $P(W < x_0) = p_0$, therefore $r(1) = (x_0 K)^{-1/\alpha}$.

Notice that if $p_0 = 1/3$, then $x_0 \approx 20$, $r(1) = (x_0 K)^{-1/\alpha} \approx 0.12$. And then that $\sigma(1) = \pi r(1)^2 \approx 0.045$.

Modelization of Fading

The propagation of radio waves in presence of random obstacles experiences random fading. Usually, modelization of fading consists in the introduction of a random factor F modeling signal attenuation at distance r : $r^{-\alpha}$. For example $\log F$ is uniform on $[-v, v]$. In this case we have a new expression of $\tilde{w}(\theta)$:

$$\tilde{w}(\theta) = \exp(-\pi\lambda\Gamma(1 - \frac{2}{\alpha})\phi(-\frac{2}{\alpha})\theta^{2/\alpha}), \tag{3.6}$$

with $\phi(s) = E(F^{-s})$, the Dirichlet transform of the fading.

Therefore, to compute $P(W < x)$, the only change in (3.5) involves a tuning of parameter C . When fading is uniform on $[-v, v]$ we have $\phi(s) = \frac{\sinh(sv)}{sv}$.

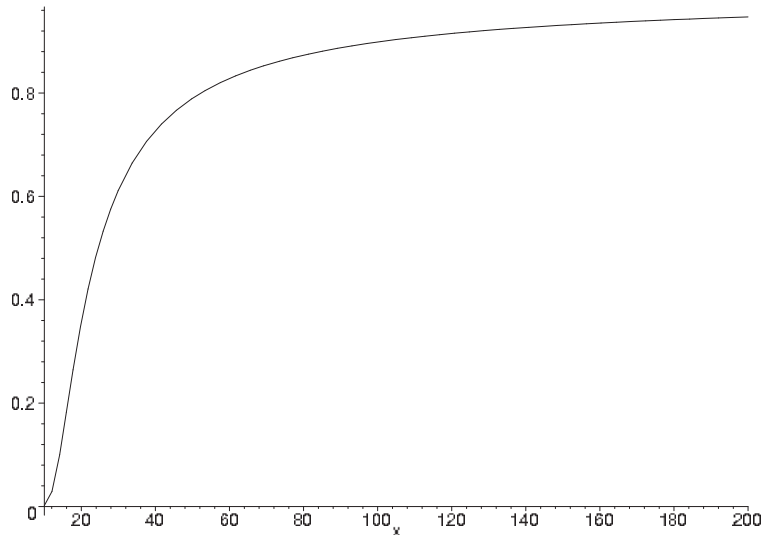


Figure 3.1: Quantity $P(W < x)$ versus x for $\alpha = 2.5$, no fading.

3.1.2 Minimizing the Number of Retransmissions

In this section we estimate the best threshold on p_0 to consider a neighbor node to really be in the neighborhood. The objective is to minimize the number of retransmissions of a packet when routed to its destination. By retransmission we mean the retransmission due to multihopping as well as the retransmissions due to packet collisions. We assume that each slot is used by unicast packets (re)transmitted *à la* ALOHA until they are correctly received by the next node.

Therefore, we want to optimize the neighborhood by excluding from it “bad” neighbor nodes that feature a too low probability of successful one hop packet transmission. They might be too far or behind an obstacle: in any case the link is not reliable enough and the number of retransmissions needed for a correct reception is not worth the hop distance. In other words, we want the best possible ratio of hop distance over number of retransmissions.

For this end we tune the parameter p_0 . The optimal value does not depend on λ but it depends on factor α , as we see below. If the probability of successful transmission is p_0 then the average number of retransmissions for one hop is $\frac{1}{p_0}$. And thus we have to optimize the quantity $p_0 r(\lambda)$, *i.e.*, $rP(W < r^{-\alpha}/K)$. All computations done for $\alpha = 2.5$ (see Figures 3.2 and 3.3) we get $\sqrt{\lambda}r(\lambda) \approx 0.089$ and we see that the optimum is $p_0 \approx 0.75$. So roughly, if a node logically excludes from its neighborhood any neighbor from which it successfully receives less than 75% of the hellos actually sent by this neighbor, we ensure a simple optimization of the overall number of retransmissions, on a network-wide level. In Figure 3.4 we depict the optimal threshold p_0 for different values

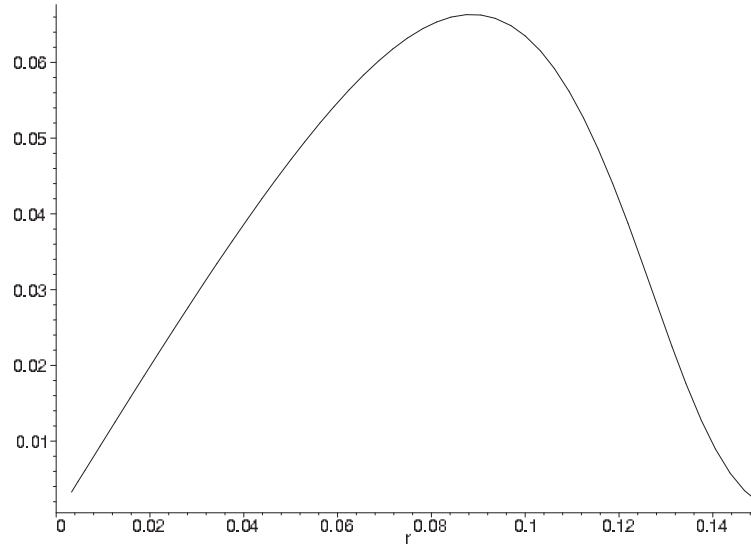


Figure 3.2: Quantity $p_0 r$ versus r for $\alpha = 2.5$, no fading.

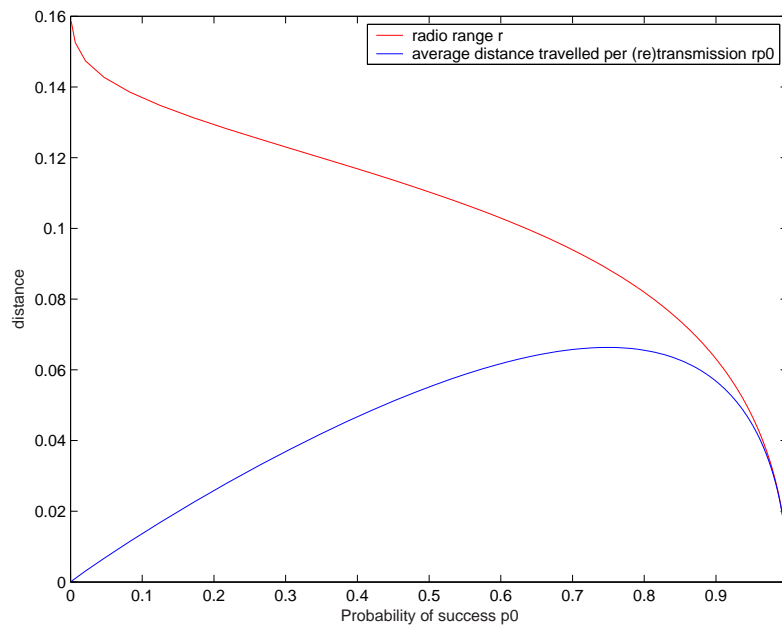


Figure 3.3: Reception range r and quantity $p_0 r$ versus p_0 for $\alpha = 2.5$, no fading.

of α . For example, we have $p_0 \approx 0.66$ when $\alpha = 4$. For the rest of this chapter, we will use $\alpha = 2.5$ for the numerical examples, thus considering the most pessimistic case.

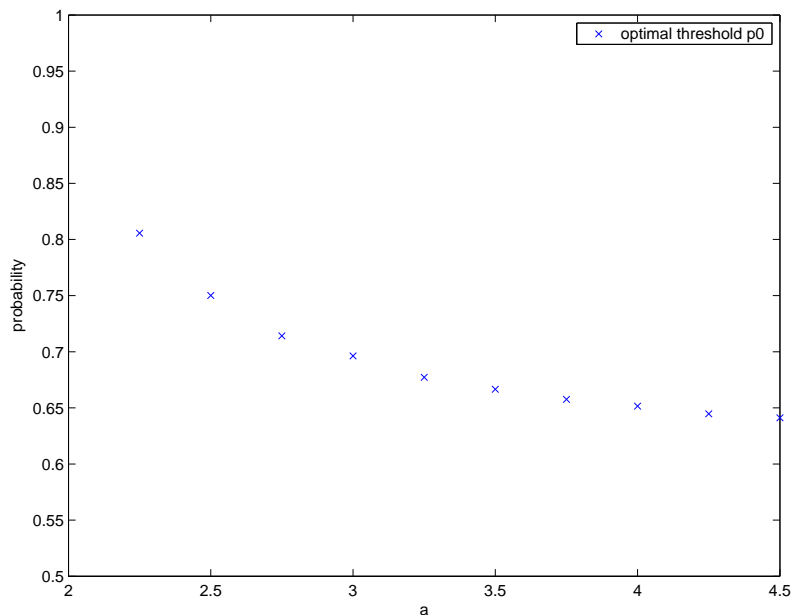


Figure 3.4: Optimal threshold p_0 versus α .

Under the Aloha-like MAC protocol considered and in conjunction with the analysis in [64] about the curvature of shortest paths in wireless networks, minimizing the number of retransmissions will result in routes with minimum average delay. This is justified because the paths will actually circumvent areas with heavy traffic. In fact, a shortest path protocol, such as OLSR can provide routes that are optimal (in a macroscopic scale) with respect to average delay, by appropriately tuning the neighborhood management. This kind of behavior verifies the experiential evidence in [87], where it was observed that it is difficult to improve the performance of OLSR using average delay based routing. This observation provides additional motivation for more elaborate delay estimations, such as the ones we proposed in Chapter 2, in order to further improve the performance and meet the QoS requirements of multimedia applications.

3.2 Scalability of Routing Protocols

Gupta and Kumar have shown in [58] that when the size of the network N increases, the neighborhood size is $O(\log N)$ and the number of hops increases at least in $O(\sqrt{\frac{N}{\log N}})$. This means that the average neighborhood size tends to be constant when the network size increases. Our model in the previous section confirms this property since it states that when the nodes are distributed over an infinite plane, the average traffic generated inside the neighborhood radius is equal to $\lambda\sigma(\lambda) = \sigma(1)$, a constant that we determined.

The neighborhood size depends on the traffic control generated by each node: the

bigger is the amount of control traffic, the smaller is the neighborhood size. Therefore, performance may vary with the use of different protocols, yielding different control traffic patterns. In this section we study more precisely the scaling properties of OLSR. An analysis which also looks into the widely used over the Internet OSPF link state routing protocol [86] can be found in [1].

3.2.1 OLSR Scalability

We will consider that the network is uniformly distributed with density ν over an area of finite size \mathcal{A} . The total number of nodes in the network is $N = \nu\mathcal{A}$. If λ is the traffic density in the network, then the average number of neighbors per node is $M = \sigma(\lambda)\nu = \sigma(1)\frac{\nu}{\lambda}$. The aim here is to derive the traffic density generated by the protocol control packets. Generally, there are two sources of control traffic: neighbor sensing on one hand, and topology discovery on the other hand.

Neighbor sensing consists in each node periodically transmitting a Hello message containing the list of neighbors heard by the node. By comparing their lists the nodes can determine the set of neighbors with which they have symmetric links. Let h be the rate (per slot) at which nodes refresh their neighbor information base and let B be the maximum number of node identifiers that a slot can contain. Let the slot duration be b seconds, which we assume to be very small. For a network with the capacity of Wifi (1-11 Mbps) we have $B = 10^5 b$. For instance, an OLSR node generates Hellos every 2 sec, *i.e.*, $h = \frac{b}{2}$. If the neighbor list exceeds B then the node generates several Hellos per update period and distributes the neighbor list among these several Hellos. The node must generate $\lceil \frac{M}{B} \rceil$ Hellos per Hello period. Therefore the Hellos lead to a traffic density of $h\nu\lceil \frac{M}{B} \rceil$. Omitting the fractional part, we get:

$$\lambda = h\nu\frac{M}{B}, \quad (3.7)$$

if the Hellos are the only source of control traffic. Since $M = \sigma(1)\frac{\nu}{\lambda}$ we get:

$$\frac{\sigma(1)}{M} = h\frac{M}{B}. \quad (3.8)$$

In fact this is only an upper bound because the network size might be smaller than $\sigma(1)$. Therefore, taking into account only the Hello control traffic, the maximum manageable neighborhood size is $\sqrt{B\sigma(1)/h} \approx 71$. This applies to any protocol that uses such Hellos.

In topology discovery, an OLSR node periodically:

1. transmits TCs with rate $\tau = b/5$. A TC contains the list of neighbors having selected the node as MPR (its MPR selectors),
2. retransmits received TCs only once (and with large jitter), and such only when the node has been selected as MPR by the neighbor from which it first received the TC.

Let M_r be the average number of MPRs selected by a node with neighborhood size M . Since the network model we consider is equivalent to a unit disk graph when we fix the neighbor threshold p_0 , it comes from [66] that $M_r \leq (9\pi^2 M)^{1/3}$. Simulations show that $M_r \sim \beta M^{1/3}$ when $M \rightarrow \infty$ with $\beta \approx 5$ (see Figure 3.5). Simulations were performed up to $M = 6,000,000$.

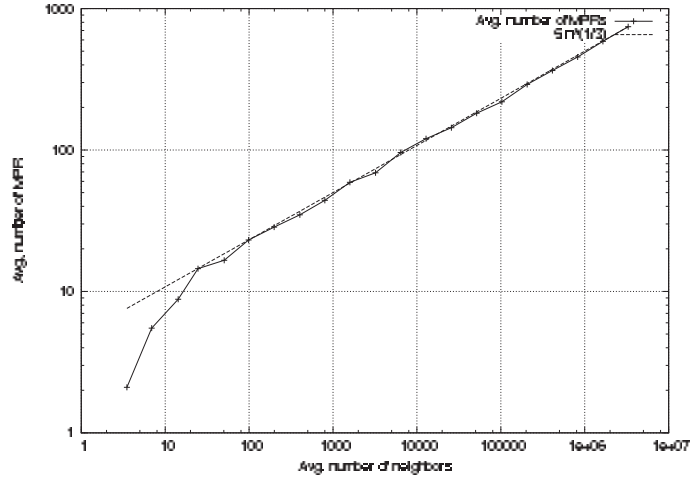


Figure 3.5: Average MPR set of a node versus neighborhood size.

In [18] it is proven that an MPR flooding costs on average $M_r N/M$ retransmissions. Therefore we get the following traffic density identity:

$$\lambda = h\nu \left\lceil \frac{M}{B} \right\rceil + \tau\nu \frac{NM_r}{M} \left\lceil \frac{M_r}{B} \right\rceil. \quad (3.9)$$

In the following, we drop the ceil factor:

$$\lambda = h\nu \frac{M}{B} + \tau\nu \frac{N}{M} \frac{M_r^2}{B}. \quad (3.10)$$

Using $M = \sigma(1)\frac{B}{\lambda}$, we have the identity:

$$\frac{\sigma(1)B}{M} = hM + \tau(M_r)^2 \frac{N}{M}. \quad (3.11)$$

This outlines a direct relation between the total size of the network N , and the average neighborhood size M . Notice that when N increases, M decreases. This corresponds to the fact that as more and more nodes are concentrated in a single radio range, interferences and collisions make more and more links perform too badly to be considered valid. Therefore more and more nodes that are theoretically directly reachable (because physically within radio range) are not considered neighbors, and hence, M decreases. The absolute minimum for M is 2, below which the network does not have a significant

connected component. If a single fully connected network is wished for, this threshold is raised to $M = \log N$.

Furthermore, the limit $M = 2$ (which in turn implies $M_r = 2$ in the worst case) yields a maximum network size for OLSR of:

$$N_{\max} = \left(\frac{\sigma(1)B}{4} - h \right) \frac{1}{\tau}, \quad (3.12)$$

which gives $N_{\max} = 3,000$.

On the other hand, when the network size decreases, it reaches a level where $N = M$. Below this level the network is only one hop (fully meshed), and the control traffic does not saturate the neighborhood. This corresponds to the maximum manageable neighborhood size. From (3.11) we get that the maximum manageable neighborhood size for OLSR is $N = 35$. Having an average neighborhood size as big as possible is important in that it reduces the average number of hops needed to go from a given source to a given destination. This way the amount of retransmissions network-wide (hence the overhead) is reduced.

We also model a slight variation of OLSR called F-OLSR, for full-OLSR. In F-OLSR the TCs contain the list of all the adjacent links, and not just MPRs. Therefore every node has the knowledge of the complete link state of the network instead of its restriction to MPR links. The TCs are still forwarded via MPR nodes. The identity for F-OLSR is then:

$$\frac{\sigma(1)B}{M} = hM + \tau M_r N. \quad (3.13)$$

It then comes that the maximum manageable neighborhood size for F-OLSR is at $N = 27$.

With Figure 3.6 we show the respective neighborhood size versus network size for the two versions of OLSR.

As the network size increases, both types of approaches feature slowly decreasing (towards 0) neighborhood size. Therefore, they fail to reach the Gupta and Kumar scalability: if the network size grows to be too big, it will break down by not being able to create significant connectivity. This is due to the fact that the topology information that each node in the network has to (re)transmit tends to increase linearly with the size of the network. This in turn yields an upper bound on the maximal size of the network, which we have computed to be of about 3,000. One way to work around this problem is to establish a hierarchical protocol that takes advantage of the scaling properties of node clustering and super-clustering. This technique greatly reduces the transit of topology information between clusters, but complexity remains in adequately distinguishing and forming different clusters. This is especially difficult in an inherently decentralized and mobile environment like ad hoc networks. However, OLSR just needs minor modifications in order to reach the Gupta and Kumar scalability. In the following section we describe the ‘‘Fish Eye’’ strategy [56]. With this strategy the overall incompressible overhead induced by periodical topology updating tends to be constant

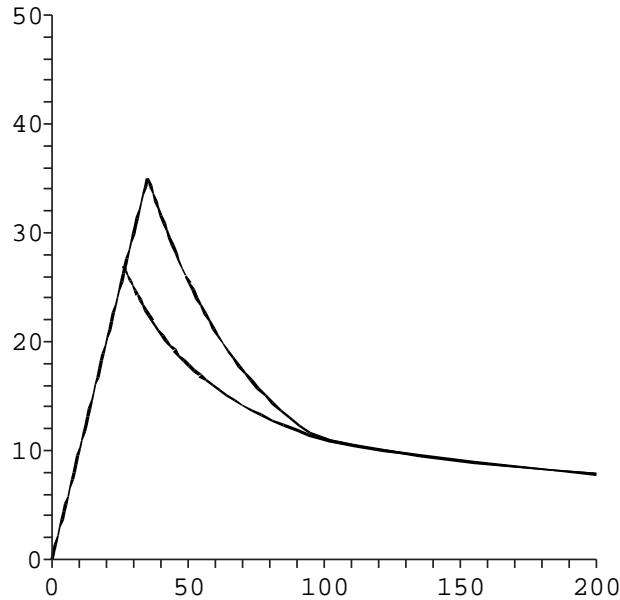


Figure 3.6: Neighbourhood size versus the network size, $\alpha = 2.5$, no fading, respectively for F-OLSR (bottom) and OLSR (top).

instead of linearly increasing with the network size. Of course this doesn't come without a cost, *i.e.*, less accurate information about the link status of remote nodes. However, this cost is not expensive: it does not degrade the delivery reliability and it does not introduce additional overhead in form of longer paths (see [36]).

3.2.2 Fish Eye OLSR

The principle of the Fish Eye strategy is that TC information from remote nodes is less frequently received, and the more remote, the less frequent. Inside the OLSR framework, nodes send TC packets with variable TTL count and VTime. The TTL limit is the maximum number of hops a packet can be relayed before being discarded and the VTime is the maximum time for which the information carried by this packet is considered valid. A node transmitting a packet with low TTL value ensures that the packet will be forwarded only inside the vicinity of this node, and not further. Conversely, a large TTL value (the maximum value is 255) ensures that the packet will be forwarded in the entire network.

Each node uses a decreasing function $f(D) \leq 1$ to determine the fraction of the TCs which are generated with a TTL larger than D (D is an integer indicating the number of hops away that the TC may reach). When no Fish Eye strategy is employed, $f(D) = 1$

for any value of D . We can assume that $\sum_{D=1}^{\infty} Df(D) < \infty$. This is indeed always the case, since $f(D) = 0$ for all $D \geq 255$. Of course, information that is received less frequently should not age as rapidly as frequently received information. This can be achieved by adequately tuning the VTime field in the TC packets.

Let us consider a node at the center of a circular network: N nodes uniformly dispatched on a disk. M is the average number of neighbors of the central node. In this case, the central node has $3M$ two hop neighbors, and $(D^2 - (D - 1)^2)M$ D -hop neighbors, for $D \leq \lfloor \sqrt{N/M} \rfloor$ (it comes that $2\sqrt{N/M}$ is the diameter of the network). The frequency of TCs received by the central node from D -hop neighbors is $f(D)\tau$. Therefore the frequency at which the central node relays TCs is

$$\tau M_r \sum_{D=1}^{\lfloor \sqrt{N/M} \rfloor} (D^2 - (D - 1)^2) f(D).$$

We will call $\phi(x) = \sum_{D=1}^{\lfloor \sqrt{x} \rfloor} (D^2 - (D - 1)^2) f(D)$. It then comes that the control traffic of the central node equals to

$$h \frac{M_r}{B} + \tau \phi \left(\frac{N}{M} \right) \frac{M_r^2}{B},$$

and we get the following general identity:

$$\frac{\sigma(1)B}{M} = hM + \tau \phi \left(\frac{N}{M} \right) (M_r)^2. \tag{3.14}$$

When $N \rightarrow \infty$ with $\phi(\infty) = 4$ we get an average neighborhood size converging towards $M \rightarrow 18$.

Figure 3.7 shows an example for function ϕ : $\phi(x) = \frac{4x}{3+x}$. Figure 3.8 shows the neighbor size evolution with respect to this function ϕ and compares it to basic OLSR.

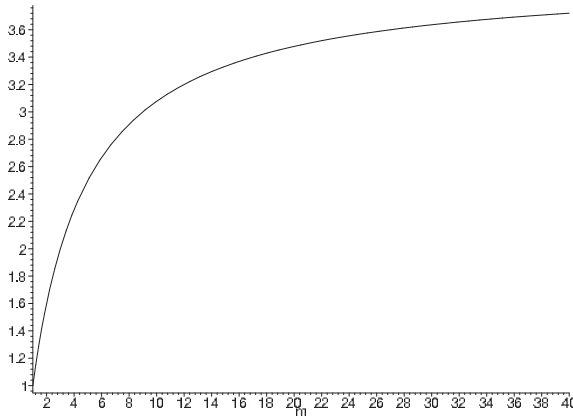


Figure 3.7: Example of function ϕ used for Fish Eye strategy.

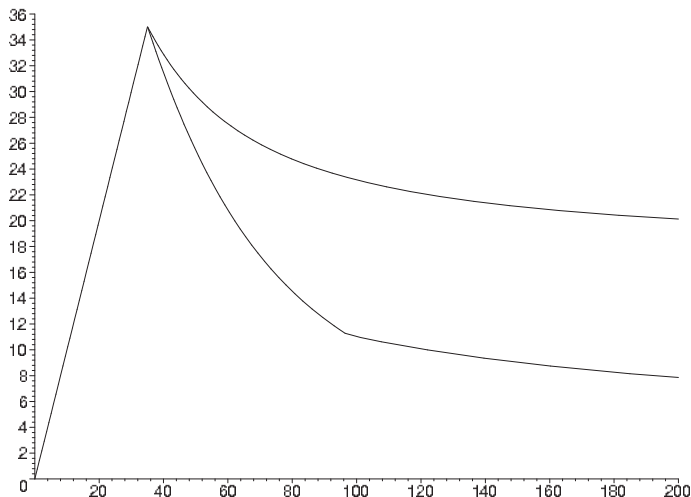


Figure 3.8: Neighborhood size versus the network size, $\alpha = 2.5$, no fading, respectively for OLSR (bottom) and OLSR with Fish Eye (top).

3.2.3 Useful Capacity

In this section we estimate the useful capacity with the OLSR protocol, as well as with Fish Eye OLSR. This quantity corresponds to the total capacity in the network available for data transfer, taking into account the protocol overhead. We denote ρ the average quantity of data traffic generated by each node. We assume that on average, the network diameter in number of hops is $\ell\sqrt{N/M}$, where ℓ denotes a linear factor that depends on the actual shape of the network area \mathcal{A} . Therefore each packet must be retransmitted $\frac{\ell\sqrt{N/M}}{p_0}$ times. This leads to an average traffic density generated by each node (including control traffic and retransmissions) of:

$$\lambda_1 = h\frac{M}{B} + \tau(M_r)^2\frac{N}{MB} + \frac{\rho\ell}{p_0}\sqrt{N/M}, \text{ for OLSR,}$$

$$\lambda_1 = h\frac{M}{B} + \tau\frac{(M_r)^2}{B}\phi\left(\frac{N}{M}\right) + \frac{\rho\ell}{p_0}\sqrt{N/M}, \text{ for Fish Eye OLSR.}$$

Therefore, using the identity $\lambda_1 = \frac{\sigma(1)}{M}$ we get an expression of ρ as a function of N and M . Clearly, for a given fixed N , ρ is maximized when M is minimized, the minimal value being $M = 2$. This yields Figure 3.9, which displays the overall maximum capacity $N\rho$ versus network size for basic OLSR and for Fish Eye OLSR (we took $\frac{\ell}{p_0} = 1$). Notice that basic OLSR with default tuning collapses at $N > 3000$, while Fish Eye OLSR features an overall capacity that keeps growing in \sqrt{N} .

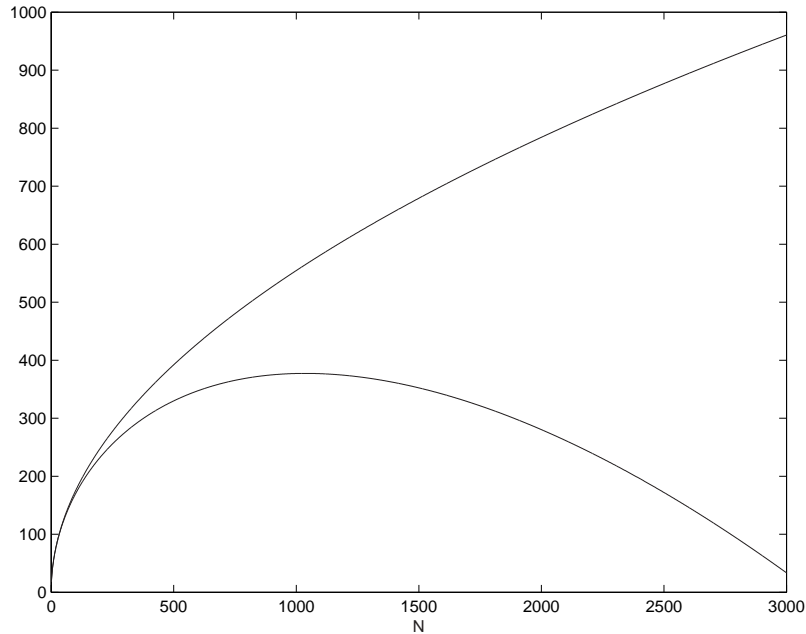


Figure 3.9: Maximum overall capacity versus the network size, $\alpha = 2.5$, no fading, respectively for OLSR (bottom) and OLSR with Fish eye (top).

3.3 Conclusion

In this chapter, we evaluated the scalability properties of ad hoc networks and routing protocols. We discussed how a local neighborhood optimization in a shortest path routing protocol like OLSR can lead to global network performance which is asymptotically optimal with respect to packet relaying. On the other hand, we showed that the nature of the routing algorithm in use impacts essentially on the maximum manageable neighborhood size, via the control traffic it induces, limiting the maximum number of manageable neighbors. As a result, there is a limit to the number of nodes in the network above which there is no significant connected component, due to incompressible topology update control traffic. We have computed this limit to be 3,000 nodes for OLSR. In fact, none of the popular ad hoc routing solutions ([37], [91], *etc.*) really scales. However, we described how link state routing can attain the famous theoretical scaling bounds outlined by Gupta and Kumar with the enhancement of Fish Eye strategies, which can be very simply incorporated into the OLSR framework. An interesting direction for further research consists in taking into account the mobility of the nodes and evaluating the performance of the Fish Eye technique in this context. In fact, it is possible to characterize the frequency at which TC messages must be sent at a given distance as a function of the nodes' maximum speed.

Appendix: Factor λ in $r(\lambda)$

By definition $\int_0^{r(\lambda)^{-\alpha/K}} w(x) dx = p_0$. Using the reverse Laplace transformation we have $w(x) = \frac{1}{2i\pi} \int_{-i\infty}^{+i\infty} \tilde{w}(\theta) e^{\theta x} d\theta$. Inserting this expression in the first equation and commuting integral signs, since $\int_0^{r(\lambda)^{-\alpha/K}} e^{\theta x} dx = \frac{e^{\theta r(\lambda)^{-\alpha/K}} - 1}{\theta}$, yields:

$$\frac{1}{2i\pi} \int_{-i\infty}^{+i\infty} \frac{e^{\theta r(\lambda)^{-\alpha/K}} - 1}{\theta} \tilde{w}(\theta) d\theta = p_0.$$

The change of variable $\lambda^{\alpha/2}\theta = \theta'$ makes λ disappear from the $\tilde{w}(\theta)$ expression:

$$\frac{1}{2i\pi} \int_{-i\infty}^{+i\infty} \frac{e^{\theta'(r(\lambda)\sqrt{\lambda})^{-\alpha/K}} - 1}{\theta'} \tilde{w}(\lambda^{\alpha/2}\theta') d\theta' = p_0.$$

Since $\tilde{w}(\lambda^{\alpha/2}\theta')$ is independent from λ and $r(\lambda)$ appears multiplied by $\sqrt{\lambda}$, we get that $r(\lambda)$ is simply proportional to $1/\sqrt{\lambda}$: $r(\lambda) = r(1)/\sqrt{\lambda}$.

Chapter 4

Multicast Scaling Properties in Large Ad Hoc Networks

In this chapter we continue the study of scaling properties and routing protocols in massive ad hoc networks in the context of multicast, *i.e.*, one to many, communication. Multicast offers an elegant way to establish group communication between users by using the concept of multicast groups, which are defined by their corresponding address. Interested *clients* can join and leave those groups in order to send and/or receive data from other group members. Moreover, the mechanisms which enable multicast communication ensure that an efficient strategy is used to deliver the data packets to all the members simultaneously. Therefore, multicast communication is adequate for a large class of applications, such as video-conferences, multi-player games, streaming applications *etc.* The previously described requirements make multicast routing an important and difficult challenge in the Internet, and even more so in ad hoc networks. In fact, mainly due to the dynamic nature of the routes, multicast protocols developed for wired networks cannot operate in the harsher mobile environment. This creates a need for protocols which are specially adapted to ad hoc networks. However, although the capacity of mobile ad hoc networks has been a very active research area since the seminal paper of Gupta and Kumar [58], the specific impact of multicast routing has not attracted too much attention, with the exception of [104]. We aim in this thesis to investigate the multicast capacity of wireless networks, and to propose a protocol solution taking into account these information theoretic considerations.

One of the advantages of multicast routing is that it reduces the total bandwidth required to communicate with all group destinations, since some links can be common to several destinations. In wired networks, the gain of multicast communication has been studied in [17, 33, 92], by estimating the ratio of the number of links in a multicast tree to n destinations over the average unicast hop distance between two random nodes. The resulting normalized multicast cost has been found experimentally to scale in $n^{0.8}$. The gain of multicast is reflected by how far the normalized multicast cost deviates from linear

growth. Except from evaluating the protocol performance, such analytical cost estimates can also be useful for the efficient management and accounting of multicast services in the network [95]. However, the topology of mobile ad hoc networks is significantly different and one would expect a much different scaling law too. Indeed the average unicast hop distance in wired networks is usually of the order $\log N$, where N is the total number of nodes in the network, while in ad hoc networks the average distance grows proportionally to $\sqrt{N}/\log N$, since the optimal neighbor degree increases in $O(\log N)$ when the capacity increases.

In this chapter, we establish performance bounds on the expected size of multicast trees as a function of the number of multicast destinations n , both via analytical methods and via simulation. In random mobile ad hoc networks, the gain of multicast communication compared to unicast is significantly larger than in wired networks. We show that a scaling law in $O(\sqrt{n})$ holds for the normalized multicast cost and, based on the analysis, we propose a protocol to be used in conjunction with the unicast routing protocol OLSR. We also show that the performance of the protocol is significantly better than MPR flooding, *i.e.*, the optimized broadcast mechanism which is already implemented in OLSR, for a vast scale of group sizes. These results can provide further motivation in supporting multicasting in mobile networks, besides the advantages of group-oriented communication. The implications of this scaling law consist in a significant increase of the total capacity of the network for data delivery, while the total amount of generated data will actually decrease (compared to the case where each node transmits data to one single destination), and both are proportional to $\sqrt{\frac{N}{\log N}}$.

The remainder of this chapter is organized as follows. In Section 4.1 we present the network model and provide analytical results on the scaling law of the normalized multicast cost. The impact of multicasting in the capacity of the network is discussed and we present measurements on multicast scaling derived from simulations in generated graph models of wireless ad hoc networks. In Section 4.2, we introduce MOST, a new multicast protocol for ad hoc networks, which is based on the previous analysis. We evaluate the performance of the protocol through simulations, which we compare to the analytical results, and we overview how the protocol was implemented for use in real network environments.

4.1 Asymptotic Multicast Properties in Ad Hoc Networks

4.1.1 Multicast Cost Scaling Law

In this section we will quantify the cost of multicast communication versus the average unicast cost. We assume that nodes have a complete knowledge of the network topology. In order to optimize the control traffic we will see in a further section how we can somewhat relax this hypothesis in the use of the OLSR link state routing protocol.

Model Description

We assume that N nodes forming a massively dense ad hoc network are distributed according to a Poisson process in an area of arbitrary size \mathcal{A} , following the model of Section 3.1.1. In this case, *i.e.*, when N is large, routes can be considered as continuous lines between nodes, and the number of retransmissions needed for a packet to reach its destination is $\Theta\left(\frac{d}{r}\right)$, where r is the typical radio range and d is the Euclidean distance from the source to the destination [64, 24]. Hence, we can represent a massively dense ad hoc network with an Euclidean graph, in which the edge costs are proportional to hop distances between nodes. The result of Gupta and Kumar [58] states that the maximum bandwidth is attained when the radio range is $r = k\sqrt{\frac{\log N}{N}}$, where k is a constant which depends on signal propagation and medium access control. A source and a multicast group of size n are chosen uniformly at random among the N nodes. We assume here that $n \ll N$. As a result, the $n+1$ multicast nodes are distributed in the area according to a Poisson process of intensity $\frac{n+1}{\mathcal{A}}$.

An optimal multicast tree is a Steiner tree, *i.e.*, a tree of minimal cost connecting all of the multicast nodes via an arbitrary subset of the remaining nodes that are not in the multicast group. Therefore, the problem of finding the optimal tree is NP-complete, even in Euclidean graphs, although in this case there is a polynomial time approximation scheme [22]. Note that since we assumed here that $n \ll N$, the wireless multicast advantage only provides an asymptotically marginal improvement. In case the number of clients n becomes significant with respect to the total number of nodes N in the network, the wireless environment would permit to further reduce the total number of retransmissions in order to reach all group destinations. This is made possible by using a connected dominated set (as is the case with MPR flooding), or other algorithms specially adapted to the wireless environment (see [106] for optimized algorithms taking also into account energy efficiency). However, although some experimental results are provided in the simulations section, we do not consider this case analytically when $n \ll N$. In any case, the upper bounds we present remain valid.

Since the problem of finding the optimal tree is intractable, we will use an approximation. We consider the two more common cases of minimum spanning trees and shortest path trees.

Minimum Spanning Trees

First, we consider multicast trees corresponding to minimum spanning trees on the $n+1$ multicast nodes. In a minimum spanning tree branching is constrained only to multicast nodes, and the computation can be performed in polynomial time. On the other hand, in Steiner trees, branching can occur on any node (or any point in the plane in the Euclidean case). In metric graphs, the cost of a minimum spanning tree is within twice the cost of an optimal Steiner tree [105]. However, it can be shown that the Euclidean minimum spanning tree is not longer than $\frac{2}{\sqrt{3}}$ times the optimal Euclidean Steiner tree [43]. Hence,

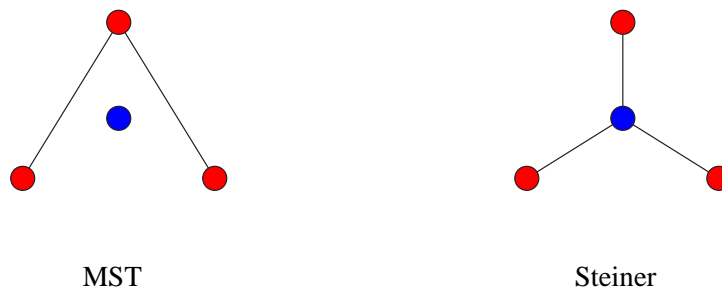


Figure 4.1: Comparison of a Minimum Spanning Tree (MST) with an optimal Steiner tree in an Euclidean graph.

in the case of massively dense ad hoc networks, minimum spanning trees yield results which are very close to the optimal. In Figure 4.1 we depict an example of a minimum spanning tree as well as a Steiner tree, in an Euclidean graph of 4 nodes. The red nodes in this example would be the clients that must be covered by the multicast tree. As we can see, the possibility of using the blue non-multicast node in the Steiner tree offers a length improvement of $\frac{\sqrt{3}}{2}$, which corresponds in this particular case to the worst case bound.

To proceed we will compute the expected path length (in meters) of a minimum spanning tree on $n + 1$ points in an area \mathcal{A} . We denote this length $L(n + 1)$.

From the analysis in [27, 98], in the 2-dimensional case, it comes that an upper bound, for the average path length (in meters) of a minimum spanning tree is

$$L(n + 1) \leq \gamma n \sqrt{\frac{\mathcal{A}}{n + 1}} \sim \gamma \sqrt{\mathcal{A}n}. \quad (4.1)$$

where γ is a constant that depends on the shape of the network domain. For a disk or a square we can set $\gamma = \frac{1}{\sqrt{2}}$.

We now define the normalized multicast cost $R(n)$ for a multicast group of size n as

$$R(n) = \frac{\text{multicast cost}}{\text{average unicast cost}},$$

where the costs are expressed in number of hops. In other words the multicast cost is the number of links in the multicast tree, and the average unicast cost is the average route length from a random source in the multicast group to a random destination in the multicast group.

We base our analysis on the observation that routes can be considered as continuous lines between nodes, and the number of hops needed for a packet to reach its destination is $\Theta\left(\frac{d}{r}\right)$, where r is the optimal radio range as stated by Gupta and Kumar.

The expected path length of the multicast tree in number of hops is $\Theta\left(\frac{L(n+1)}{r}\right)$, while

the average unicast cost is $\Theta\left(\frac{L(2)}{r}\right)$. This implies that, for the normalized multicast cost, it holds

$$R(n) \simeq \frac{L(n+1)}{L(2)}. \quad (4.2)$$

Quantity $L(2)$ is highly dependent on the shape of the network domain, but is of order $\sqrt{\mathcal{A}}$ when the network domain shape stays within some reasonable model. In all rigor we have $L(2) = \beta\sqrt{\mathcal{A}}$. For the disk we have $\beta = \frac{128}{45\pi^{3/2}} \approx 0.51$, for the square it becomes $\beta = \frac{1}{15}(2 + \sqrt{2} + 5 \log(1 + \sqrt{2})) \approx 0.52$.

Combining (4.1), (4.2), we get

$$R(n) \leq \frac{\gamma n}{\beta\sqrt{n+1}} = O(\sqrt{n}). \quad (4.3)$$

Hence, we obtain the multicast scaling law $R(n) = O(\sqrt{n})$. It comes that the gain of multicast over unicast, which is reflected by how far $R(n)$ deviates from linear growth, is also $O(\sqrt{n})$. This result is in contrast with similar comparisons in wired networks [17, 33, 92] where the gain of multicast communication is significantly smaller. In that case, the multicast cost scales, according to experimental studies, following a power law with exponent between .8 and .9.

More generally, we can show, using the same approach, that for a network spanning on a domain in dimension D

$$R(n) = O\left(n^{1-\frac{1}{D}}\right).$$

In [98] it is shown that the length of minimum spanning trees on points randomly placed in a hypercube is $O\left(n^{1-\frac{1}{D}}\right)$, even when the point distribution is not uniform, with some mild constraints. This implies that the multicast scaling law still holds when the multicast nodes are not distributed uniformly among the nodes of the network.

Shortest Path Trees

A popular approach in building multicast trees in wired networks consists in pruning shortest path trees rooted at the source node. In this case, we cannot prove worst case bounds on the total cost of shortest paths trees, compared to the cost of optimal Steiner trees. However, in practice, shortest path trees achieve a satisfactory performance. Moreover, shortest path trees minimize the maximum path length from the source to any destination. In the currently considered model of mobile ad hoc networks, when $n \ll N$, a shortest path tree is equivalent to n unicasts, since the expected number of branching nodes is very small. Hence for a small number of destination nodes, the gain of multicast communication is negligible.

On the other hand, when $n \rightarrow N$, the total number of hops in the tree also tends to N , since we consider a tree spanning on almost all the nodes. The average unicast

distance in hops is $O\left(\frac{1}{r}\right)$, where the radio range $r = \alpha\sqrt{\frac{\log N}{N}}$. Hence, the normalized multicast cost tends to

$$R(N) = O(Nr) = O(\sqrt{N \log N}).$$

This is the expected behavior for any method used to construct a tree spanning on all the nodes of an ad hoc network with unit cost links. Consequently, for large multicast group sizes we still expect to observe a multicast scaling law of the form $R(n) = O(n^{\frac{1}{2}+\varepsilon})$, for any $\varepsilon > 0$. In Section 4.1.3, we study the normalized multicast cost of shortest path trees experimentally.

4.1.2 Capacity of Multicast Communication

In this section we investigate the impact of the multicast cost scaling law in the capacity of the network, when nodes communicate with multicast. We are interested in the order of magnitude of the maximum attainable bandwidth. We show that similar bounds to the ones described in [104] can be obtained without the need of particularly complex additional routing mechanisms.

In presence of traffic density of λ bits per time unit per square area unit, the typical radius of correct reception r decays in $O\left(\frac{1}{\sqrt{\lambda}}\right)$ [58, 64]. If C is the capacity generated by each node, the density of traffic generated per square area unit is $\Theta(CN)$. The maximum bandwidth attainable for unicast traffic is $C = O\left(\frac{1}{\sqrt{N \log N}}\right)$.

We have shown that each multicast packet in a group of size $n \ll N$ will be retransmitted $\Theta(\sqrt{n}\frac{1}{r})$ times. This yields a traffic density (including retransmissions) $\lambda = \Theta(CN\sqrt{n}\frac{1}{r})$. Therefore

$$\begin{aligned} r &= O\left(\frac{1}{\sqrt{\lambda}}\right) = O\left(\sqrt{\frac{r}{CN\sqrt{n}}}\right) \\ \Rightarrow C &= O\left(\frac{1}{rN\sqrt{n}}\right). \end{aligned}$$

As a result, the maximum rate at which a node can transmit multicast data is $O\left(\frac{1}{\sqrt{nN \log N}}\right)$ and it is obtained for the minimum $r = O\left(\sqrt{\frac{\log N}{N}}\right)$ ¹. In this case, the total rate at which data is received by the n destinations in the multicast group is $O\left(\sqrt{\frac{n}{N \log N}}\right)$.

When all nodes in the network communicate in unicast (each node with one single destination), the total capacity of the network increases with network size in $O\left(\sqrt{\frac{N}{\log N}}\right)$.

¹As discussed in Chapter 3, the $\log N$ factors can be dropped if we assume optimally placed nodes, or if we relax the network connectivity requirement to the existence of a giant component.

However, when there are $O(N)$ nodes in the network acting as multicast sources in groups of size n (e.g. in teleconferences between n users), the total rate at which data is transmitted in the network is $O\left(\sqrt{\frac{N}{n \log N}}\right)$. Similarly, the total rate at which data is received is $O\left(\sqrt{\frac{nN}{\log N}}\right)$. Hence, compared to unicast traffic, multicast traffic results in an increase by a factor $O(\sqrt{n})$ of the capacity of the network (and per node) for receiving data, although the total distinct data transmitted will in fact decrease by the same factor.

4.1.3 Numerical Results

Comparison with Unicast

In this section, we present simulations that verify the theoretical results on generated graph models of mobile ad hoc networks. We measure the normalized multicast cost $R(n)$ for various sources and multicast groups, and take the average for each group size n . The results are plotted in log log scale, and compared to a line which corresponds to the predicted asymptotic growth. The graphs are generated by placing nodes randomly in a square for 2-D networks and in a cube for 3-D networks, and then connecting the nodes which are in a distance smaller than the critical radius for connectivity r , such that the average number of neighbors for each node is $\log N$.

In Figure 4.2, we present results corresponding to minimum spanning trees. The algorithm used to construct the minimum spanning trees will be presented in the following section, when we describe a protocol which can achieve these performance estimates. The measured cost is compared to the function $\frac{n\sqrt{2}}{\sqrt{n+1}}$, obtained from (4.3) by setting $\gamma = \frac{1}{\sqrt{2}}$ and $\beta = 1/2$, which corresponds to the approximate case where we do not consider border effects.

Figure 4.3 depicts measurements of the normalized cost of shortest path trees. The multicast cost $R(n)$ is compared to function $2\sqrt{n}$ (where the constant 2 was chosen empirically). Observe that the cost is always higher than in minimum spanning trees, although the plot grows linearly with a slope close to 0.5 for large n .

In the case of 3-dimensional ad hoc networks, for both minimum spanning trees and shortest path trees, the normalized multicast cost scales in $O(n^{\frac{2}{3}})$.

Comparison with MPR Flooding

In this section we compare the multicast overlay trees with MPR flooding, *i.e.*, the optimized broadcast mechanism which is already implemented in OLSR and takes advantage of multi-point relay nodes (*cf.* Section 1.3.2). The performance of MPR flooding was already discussed in Section 3.2.1, while detailed performance studies can be found in [76, 65]. This comparison is of interest because, although using overlay multicast

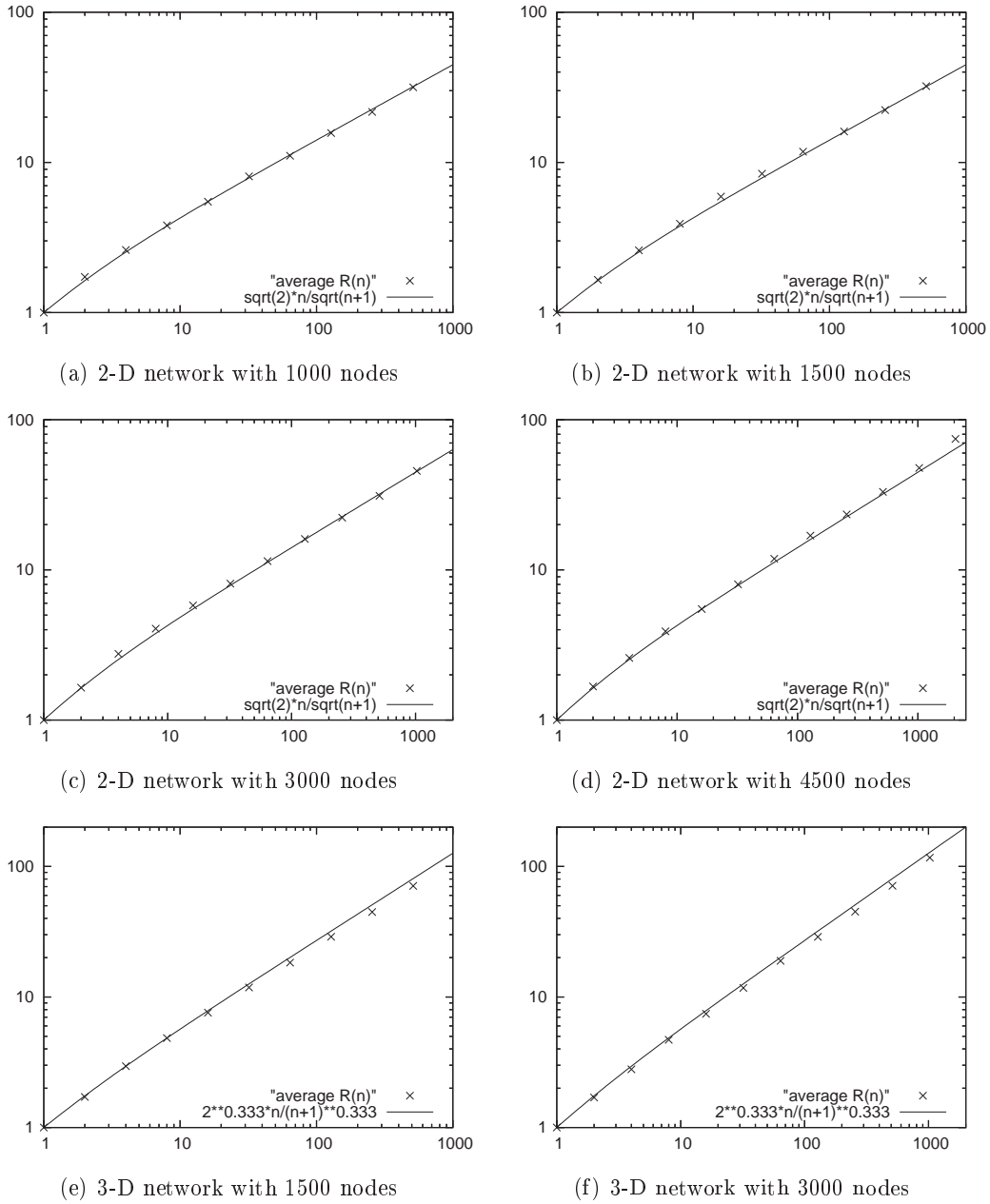
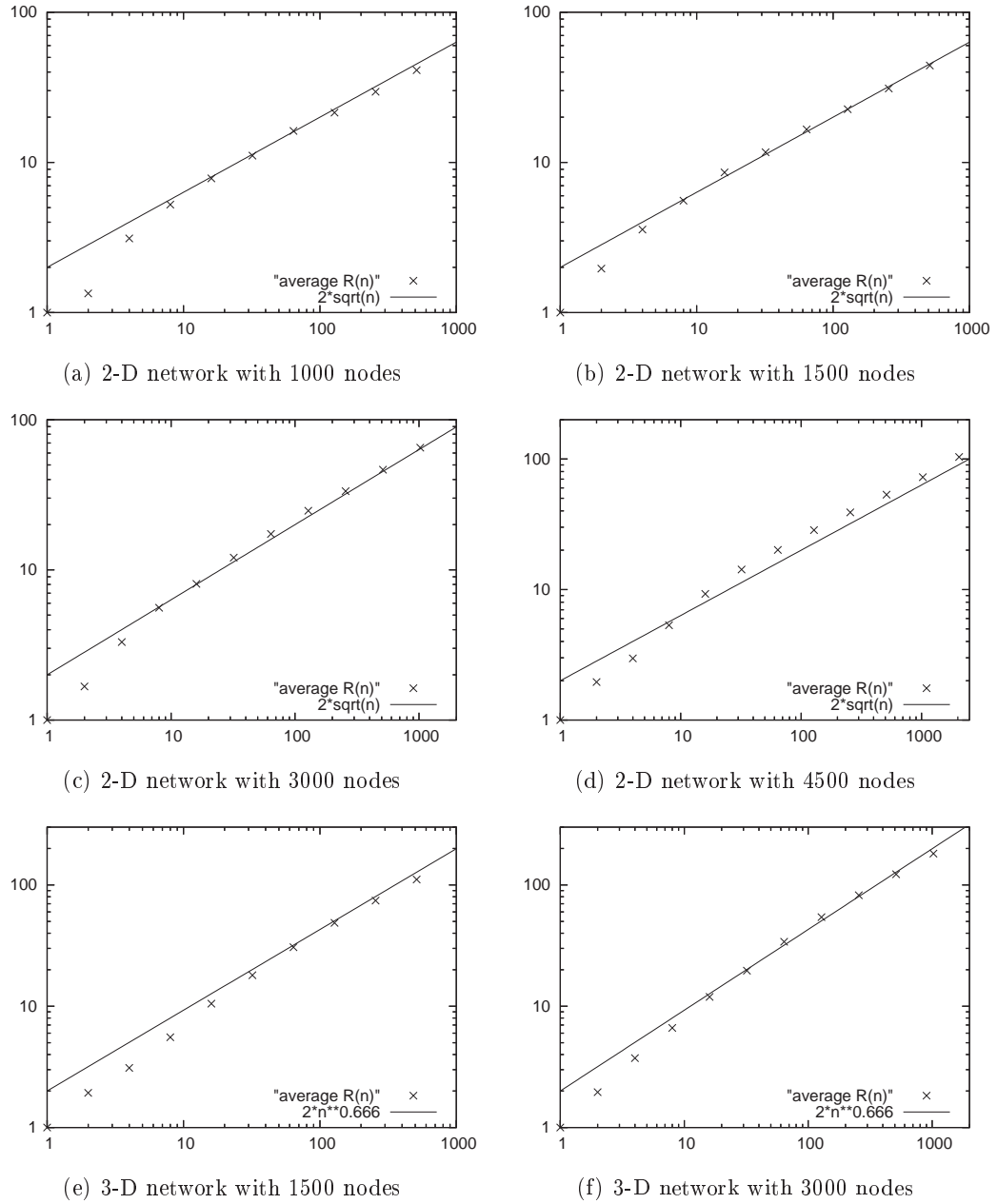


Figure 4.2: Multicast cost $R(n)$ versus multicast group size n .

Figure 4.3: Shortest path tree cost $R(n)$ versus multicast group size n .

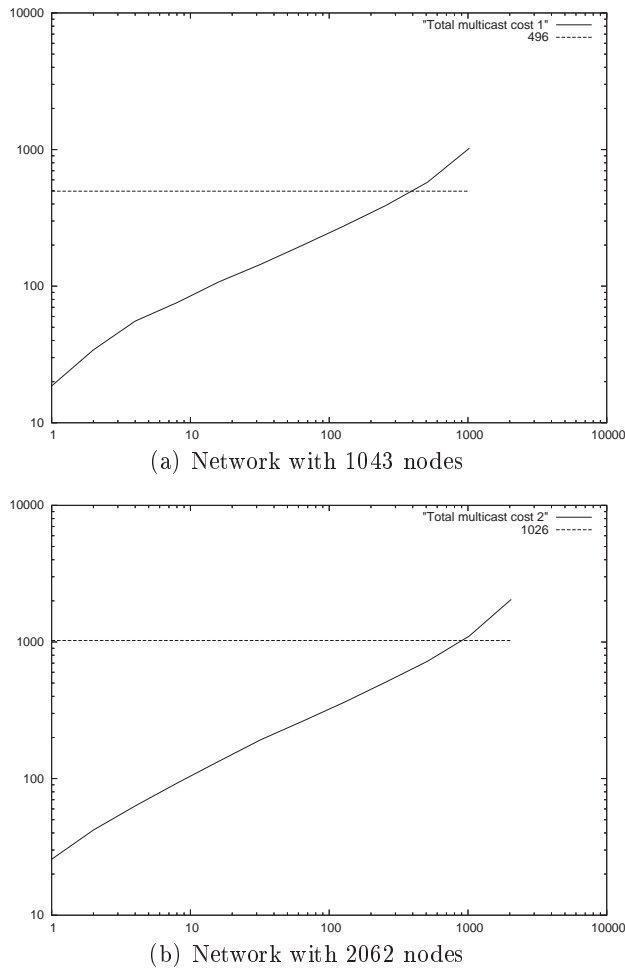


Figure 4.4: Total number of multicast retransmissions versus group size n , compared with the number of retransmissions using MPR flooding (straight lines).

trees achieves significant performance gains, this would happen at the cost of some extra protocol complexity, which could be avoided by using the existent MPR flooding technique. Hence, it is important to identify in which situations such a compromise is justified.

We perform simulations in ad hoc networks generated in the same manner as in the previous section. We measure the total multicast cost, *i.e.*, the total number of forwarding retransmissions needed to reach all the group destinations, for various sources and multicast groups, and take the average for each group size n . These measurements are compared with the average number of retransmissions that are generated using MPR flooding, initiated from the same source nodes as before. In this case, the number of retransmissions is independent of multicast groups and their size. The algorithm used

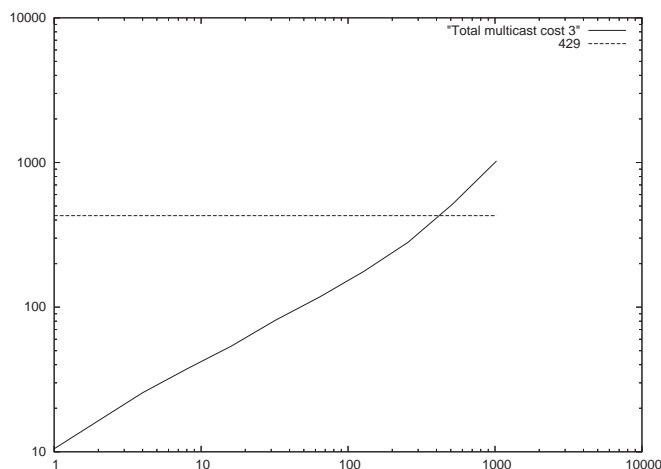


Figure 4.5: Total number of multicast retransmissions versus group size n , compared with the number of retransmissions using MPR flooding in a denser network (with 1025 nodes).

for the selection of MPR nodes is the greedy algorithm described in [76, 65].

In Figure 4.4, we present results obtained from simulations in two sparse networks of different sizes where the critical radius for connectivity r , is such that the average number of neighbors for each node is $\log N$. In this case, the number of retransmissions in MPR flooding corresponds to approximately half the nodes in the network and is depicted by the straight lines in the graphs. However, multicasting performs better in almost the entire range of possible group sizes, except for multicast groups that constitute a large portion of the network.

In Figure 4.5, we repeat the simulations in a denser graph where the average node degree is twice the number of neighbors corresponding to the critical connectivity limit. Suggestively, the average node degree in Figure 4.5 is approximately 14, while in Figure 4.4a it is approximately 7. Note that this does not influence the comparative advantage of multicast over the same range of group sizes as in sparse graphs.

4.2 Specification and Simulation of MOST Protocol

In this section, we present the Multicast Overlay Spanning Tree (MOST) protocol, in which we take into consideration the previously derived results. As discussed earlier, multicast protocols proposed for wired networks are not adapted to ad hoc networks, because of the frequent changes in tree structure due to the dynamic network topology, in addition to the group membership changes. Hence, multicast ad hoc routing is a challenging research domain, and many possible approaches have been proposed in

the research literature [39]. Multicast ad hoc protocols can be classified according to the underlying routing structure to tree-based protocols and mesh-based protocols. The routing structure can be either group shared, or source dependent. Some tree-based protocols are MAODV [94] which is an extension to the unicast routing protocol AODV [91] based on a group shared tree, MOLSR [75] which is an extension to OLSR unicast routing protocol based on a Dijkstra tree, and Adaptive Demand-driven Multicast Routing Protocol (ADMR) [69]. A protocol which is based on overlay trees is AMRoute [80]. As an example of mesh-based routing protocols we mention On-Demand Multicast Routing Protocol (ODMRP) [77] and Core-Assisted Mesh Protocol (CAMP) [54].

In contrast to these solutions, we propose a protocol aiming at achieving the theoretical capacity bounds derived in the previous section, being based on overlay group shared trees. In the next section, we describe the algorithms used by the protocol in order to maintain the multicast overlay spanning trees.

4.2.1 Overlay Tree Construction

As we saw previously, it is more efficient to consider minimum spanning trees. We discuss here Algorithm 1, which achieves the multicast cost $R(n) = O(\sqrt{n})$ calculated in Section 4.1.1. The algorithm does not require any more information than what is provided by a link state unicast routing protocol, like OLSR.

Algorithm 1 Basic Minimum Spanning Tree Algorithm

Input: Network graph.

Output: Overlay tree.

1. Find shortest paths between all pairs of multicast nodes.
 2. Build complete graph on multicast nodes with costs $c_{ij} = \{\text{length of shortest path between } i \text{ and } j\}$.
 3. Build minimum spanning tree on the complete graph, rooted at the source node.
-

The construction of the minimum spanning tree (step 3) can be implemented using Prim's algorithm [40]. The resulting tree is an overlay multicast tree, since it consists only of multicast nodes and its links are in fact tunnels in the actual network. Multicasting is achieved when each node forwards multicast packets to its successors in the overlay tree. It must be noted that in a fully distributed protocol, each node must be able to compute the same minimum spanning tree independently from the others. Therefore, we impose an ordering to the multicast nodes based on their IP addresses when executing Prim's algorithm. The computed minimum spanning tree will be directed and rooted to the node with the smallest IP address. However, this fact

has no practical importance in the protocol's operation, where the tree will be treated as a shared tree with no root. Step 1 corresponds to n Dijkstra algorithm iterations. Therefore, the total complexity is $O(n(M + N \log N))$, where n is the multicast group size, N and M are the number of nodes and edges in the network, respectively. The algorithm's expected complexity can be improved because it is not necessary in practice to compute all shortest paths from each node to all other nodes to build the minimum spanning tree.

We propose Algorithm 2 as a faster alternative to compute minimum spanning overlay trees. The algorithm is essentially equivalent to Algorithm 1, but the shortest paths are calculated in conjunction with the minimum spanning tree. Hence, it is not necessary to compute shortest paths between all pairs of multicast nodes. In fact, according to tests in wireless network topologies, this algorithm has an average running time comparable to a Dijkstra algorithm, even when the number of clients increases.

We denote $G(V, E, w)$ the network graph, where V is the node set, E is the edge set, and each edge e is associated with a cost $w(e)$. We also denote S the set of multicast nodes. The array d associates each node with a distance to the multicast overlay tree, *i.e.*, $d[v]$ corresponds to the minimum distance of node v to the multicast nodes that are already part of the tree. This distance is initialized to 0 for the root node and to ∞ for all other nodes. The array π associates each node with a predecessor multicast node. When this table has been computed, it contains the information needed to represent the overlay tree, since each multicast node will be associated with another multicast node (except from the root). The predecessors of the other nodes in the graph need only be maintained during the computations.

The algorithm manages a set F of multicast nodes that have not been covered yet by the tree, and a min-priority queue Q which includes all nodes, with the priority attribute being equal to their distance d . In each iteration the algorithm chooses a node with the smallest distance to the overlay tree (step 6), and checks whether it is a multicast node (step 7). In this case, the node's distance is updated to 0 (because the node is added to the overlay tree) and it is removed from the set F . Afterwards, for each chosen node, steps 11–15 check its adjacent nodes on whether their distance can be improved, and update the predecessors appropriately, similarly to Dijkstra's algorithm. However, in this case there are two important differences: the overlay predecessors can only be multicast nodes, hence steps 14–15 perform an additional check; moreover, previously extracted non-multicast nodes might be re-inserted in the priority queue in case their distance to the tree has improved due to the addition of new overlay nodes (therefore, the algorithm's worst case complexity remains the same as in Algorithm 1). The iteration ends when multicast nodes have been covered, hence the improvement in the average case complexity. For simplicity, we did not include the code based on the IP address ordering of the nodes. For instance, the root node here will correspond to the smallest IP address, and the priority queue will take the IP address ordering into consideration.

Due to the fact that the distance between two nodes in number of hops is proportional to the Euclidean distance, the resulting multicast tree can be considered

Algorithm 2 Efficient Minimum Spanning Tree Algorithm

Input: Weighted Graph $G(V, E, w)$, Multicast Node Set S , Root Node s .

Output: Predecessor Table π .

1. for all $(v \in V)$ { $d[v] \leftarrow \infty$; $pred[v] \leftarrow NIL$; }
 2. $d[s] \leftarrow 0$;
 3. $Q \leftarrow V$;
 4. $F \leftarrow S$;
 5. while $(F \neq \emptyset)$ {
 6. $u \leftarrow \text{EXTRACT-MIN}(Q)$;
 7. if $(u \in S)$ {
 8. $d[u] \leftarrow 0$;
 9. $\text{DEL}(F, u)$;
 10. for each $(v \in \text{adj}[u])$ {
 11. if $(d[v] > d[u] + w(u, v))$ {
 12. $d[v] \leftarrow d[u] + w(u, v)$;
 13. if $(v \notin Q)$ { $\text{INSERT}(Q, v)$; }
 14. if $(u \in S)$ $\pi[v] \leftarrow u$;
 15. else $\pi[v] \leftarrow \pi[u]$; } } }
-

as an approximation of the Euclidean minimum spanning tree on the multicast nodes.

The advantage of this approach is that only the multicast nodes need to participate in the construction of the multicast tree, while the other nodes serve merely as relays and are not necessarily aware of the multicast communication. This fact facilitates the development of a peer to peer protocol which can be run only by the participating multicast nodes, hence it could be downloaded dynamically by a node whenever it decides to join a multicast communication. However, a practical implementation of a protocol using this approach must still face the difficulties introduced by the nodes' mobility. The protocol should provide a mechanism for nodes to communicate reliably among them their participation in the multicast communication. In the following section, we discuss how these problems are treated in the MOST protocol that we propose.

4.2.2 Specification of MOST Protocol

We now present a new distributed multicast routing protocol based on group shared trees. The protocol must be used in conjunction with a link state protocol, hence we choose to develop an extension to OLSR.

In a fully distributed spanning tree design, the tree computation is performed independently by each group member. To proceed to the correct computation of the overlay tree, multicast nodes need to know the membership of their multicast group. Therefore, when a node wants to join or leave a group, it broadcasts a Join or Leave message to the entire network, via the optimized MPR-flooding mechanism used in OLSR. Join messages are sent periodically according to the *Join_Interval*, which is set by default to be equal to the TC message interval, *i.e.*, 5 seconds. The total protocol overhead is limited in these messages, independently of the number of groups and sources. In fact, MOST is well suited for managing numerous groups of small size, with arbitrary sources. Each group member must compute periodically the multicast tree to discover and maintain its overlay neighbors. In order to compute the multicast tree, the node needs information about network topology which is delivered by OLSR. The overlay neighbor set is a subset of clients sharing the same tree with that member, which are linked to it via unicast tunnels. The distance between overlay neighbors can be of one or several hops. Each client receives/retransmits multicast data from/to its overlay neighbors. The computed tree is a group shared tree, hence it must always be the same for all the clients. However, because of changes in the network topology and group membership, there is no guarantee that all clients hold the same tree. Consequently, to avoid loops there is a need to maintain a duplicate table in each client node. Moreover, some redundancy is introduced in data forwarding after tree updates to avoid packet losses.

Managing Join and Leave Messages

The Join message contains the multicast group(s) address(es) that the sender has joined. The message format is the OLSR message format described in Section 1.3.2. The sender's address is included in the *Originator* field of the message header. The Join message content format is depicted in Figure 4.6.

Each multicast node maintains a membership table with the members of all the groups it belongs to. Upon receipt of a new Join message, each concerned node adds the new client to its membership table. The entries in the membership table are also associated with an expiration time, which is determined by the *VTime* field in the Join message header. After this time period the entries are removed, unless another Join message is received, in which case the expiration time is updated.

When a node wants to leave a group, it broadcasts a Leave message to the entire network but keeps acting as a group member if it receives data for a predefined transition period. Beyond this period, received packets are not retransmitted. Leave messages

```

0                               1                               2                               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                               MULTICAST GROUP ADDRESS                               |
+-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                               MULTICAST GROUP ADDRESS                               |
+-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
:                               (etc.)                               :
```

Figure 4.6: Join message content format.

follow the same format as Join messages, hence the message contains the group addresses that the originator wishes to leave.

We note that in case there is a multicast node failure or disconnection from the network, this event will be accounted for in the OLSR topology table. Therefore, other multicast nodes will act accordingly, as if the problematic node had left all multicast groups, and the multicast communication will not be affected.

Tree Computation and Maintenance

For each group, each client computes periodically the corresponding overlay tree, according to its membership table. Therefore, the client needs to maintain a table with its overlay neighbors. The update period is set by default to the Hello interval, *i.e.*, 2 seconds, which has been found empirically to provide optimal performance. If a new node joins the group or a client leaves the group, the tree is updated immediately. Whenever the overlay neighbors change after an update, the client considers both new, as well as older overlay neighbors for a transition period of 1 second, that is half the update period. The transition period is introduced to make it possible for all the clients to take into account tree changes, and to improve the packet delivery ratio. After that period, only new neighbors are considered. Finally, in order to be able to perform the overlay tree construction according to the previously described algorithm, nodes switch to full-OLSR mode and start advertizing their complete neighbor set whenever they join a multicast group. The reason for that is that it must be possible to compute the distance between each pair of group members.

Transmission and Forwarding of Data Packets

Unlike common multicast protocols where data packets are transmitted in a broadcast mode, in MOST data packets are encapsulated in unicast packets before being forwarded to the overlay tunnels. Unicast transmissions present important additional benefits when the subnetwork layer in use is IEEE 802.11, as is the case in most actual wireless networks. Firstly, the packet delivery ratio is significantly improved, since packets are

retransmitted in case of collisions, while this is not the case for broadcast (or multicast) packets (*cf.* Section 2.1). Secondly, in most 802.11 variants (including b and g), the broadcast frames are transmitted by default at a lower rate than unicast frames. For instance, when 802.11b operates at a unicast data rate of 11 Mbps, the default broadcast rate is 2 Mbps, and most wireless card drivers do not offer the possibility to change it. Therefore, unicast tunnels can actually increase the available bandwidth.

When a client receives a multicast data packet², it checks whether the packet has already been received. If this is the case, the duplicate packet is dropped. Otherwise, the client forwards the packet to each of its overlay neighbors, except the one from which the packet was received. Source nodes act also as multicast group members, hence they simply send their data in unicast to their overlay neighbors.

4.2.3 Implementation Overview

In this section we outline our complete implementation of the MOST protocol for Linux. An overview of the architecture is depicted in Figure 4.7. The implementation consists of two modules: MDFP and OOLSR.

MDFP (Multicast Data Forwarding Protocol) is a forwarding protocol that enables point to multipoint data transfer. Multicast packets are captured and encapsulated in order to be forwarded inside a multicast tree. This module was developed for use with the MOLSRL multicast protocol [12], and we adapted it to also support MOST. OOLSR (Object oriented OLSR) is INRIA's implementation of the OLSR protocol in C++ [13]. The core of MOST was implemented as an extension inside this module.

The OOLSR module with MOST extension is in charge of sending and processing Join and Leave messages, as well as computing and maintaining the overlay multicast tree, based on the network topology. The MDFP module is in charge of the actual forwarding of multicast data packets in the overlay tree. For this purpose, it performs encapsulation and decapsulation of data packets, and maintains a table in order to detect duplicate packet receptions.

As shown in Figure 4.7, the two modules constantly exchange information. The OOLSR daemon provides MDFP with up to date overlay neighbors information, which is all that is needed to perform the transmission and forwarding of multicast data. Conversely, MDFP communicates to the OOLSR daemon the group membership information concerning the node's OLSR interfaces. In fact, multicast client applications update the interfaces' IGMP information (*cf.* Internet Group Management Protocol [47]), and this information is interpreted by MDFP. Incoming multicast data packets are captured by the *netfilter* module in the kernel, following predetermined rules (such as IP in IP encapsulation and a predetermined UDP port number). MDFP decapsulates the packets and passes them to the client applications, while it re-encapsulates them in order to forward them to the overlay neighbors. Similarly, data transmitted by a local multicast

²In fact all packets are received in unicast, so we refer here to the encapsulated multicast content.

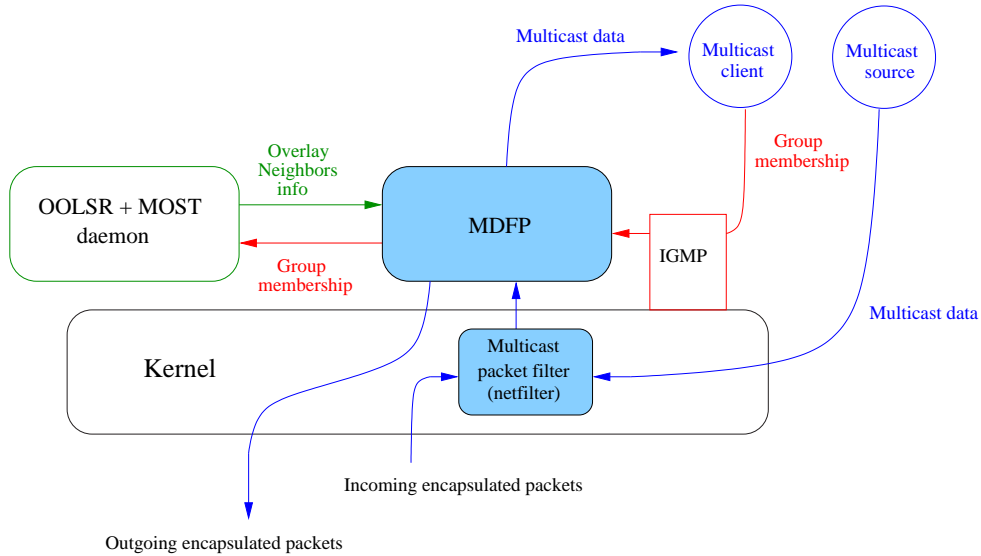


Figure 4.7: Overview of multicast implementation.

source is also captured by netfilter and processed by MDFP.

Finally, we note that the OOLSR module (including the MOST extension) can be loaded as a plugin in ns-2, hence the simulator shares the same source code as the real implementation.

4.2.4 Simulation Results

In this section we perform ns-2 simulations in various scenarios aiming mainly to verify the theoretical analysis, as well as to evaluate the protocol performance in a mobile ad hoc network environment. In Table 4.1, we summarize the parameters that are common for all our simulations.

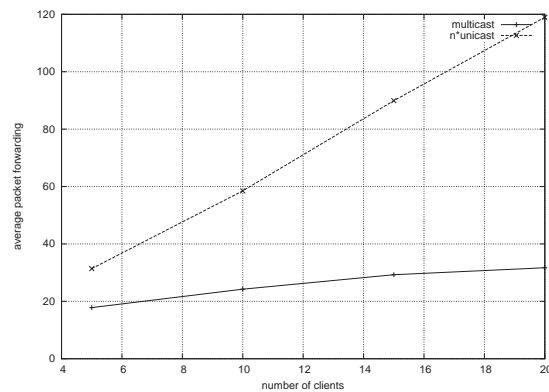
Table 4.1: Common simulation parameters.

MAC Protocol	IEEE 802.11b
MAC Rate	11Mb
Propagation model	Two ray ground
Transmission range	250m
Packet size	1200 bytes
Traffic type	CBR
Number of iterations	5

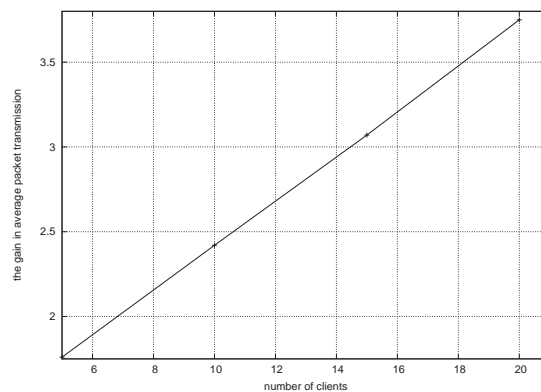
Comparison of Multicast and Unicast Performance

In Section 4.1.1 we performed an analytical comparison of multicast performance compared to unicast. We now repeat the comparison using ns-2 simulations of a fully functional protocol. We measure the average multicast cost for various group sizes, by counting the total number of times each packet is relayed to reach all destinations, using MOST. The unicast cost is determined in the same manner, by repeating the same simulations and considering OLSR unicast transmissions between each source-client pair.

The simulation environment consists of a randomly generated topology of 100 wireless nodes forming an ad hoc network, in an area of $1500m \times 1500m$. We consider group sizes ranging from 5 to 20 nodes (not including the source). Multicast groups and sources are chosen at random. The source node sends to the group CBR traffic of 64kbps, with packets of 1200 bytes, for 150 seconds of simulated time. To obtain reliable results, simulations are conducted several times with 5 different seeds. The mean results are depicted in Figure 4.8.



(a) Average packet retransmissions.



(b) Gain of multicast routing.

Figure 4.8: Comparison of multicast versus unicast to all destinations.

The analysis in Section 4.1.1 allows us to compute an upper bound on the number of multicast packet retransmissions as a function of the number of group clients n . Namely, it comes from (4.3) that an upper bound for this parameter is: $\sqrt{2n} \times d_u$, where d_u is the average unicast distance between two nodes in hops. In Figure 4.9, we compare the average number of retransmissions measured through simulations to the theoretical bound. Although the analysis is performed in an asymptotic setting, we notice that the upper bound is also valid in these simulations.

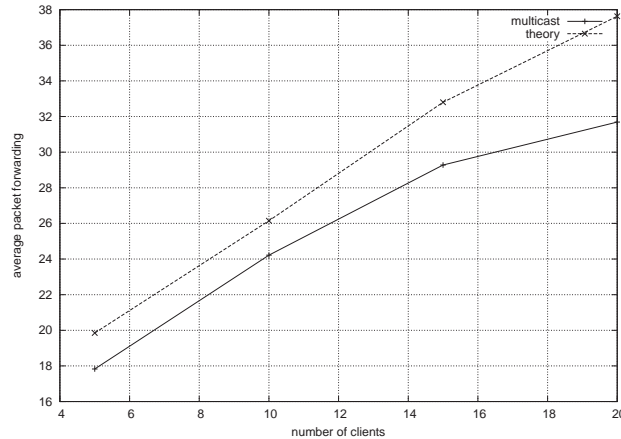


Figure 4.9: Simulation results vs theoretical upper bound.

Multicast Performance versus Throughput and Group size

Simulations are conducted to determine the maximum source rate that maintains an acceptable delivery ratio in a multicast group. By acceptable we mean that its value is higher than 95%. We consider static topologies again since the goal is to find the saturation point of the network. We consider a 200 wireless nodes network in a $1800m \times 1800m$ area, with one multicast group. We vary the number of clients as well as the source bit rate and we measure the packet delivery ratio, as shown in Figure 4.2.4. We notice that the source node can transmit with a rate of up to 200kbps with a delivery ratio higher than 99%. From a 250kbps rate, the performance remains good for small groups but it decreases for large group sizes.

We also run other simulations by fixing the number of clients to 10 and varying the number of active multicast groups in a 300 nodes network. In each group a source is transmitting CBR traffic with a rate of 64kbps. As shown in Figure 4.2.4 we notice that the delivery ratio is very high until network saturation, with 8 groups of 10 clients.

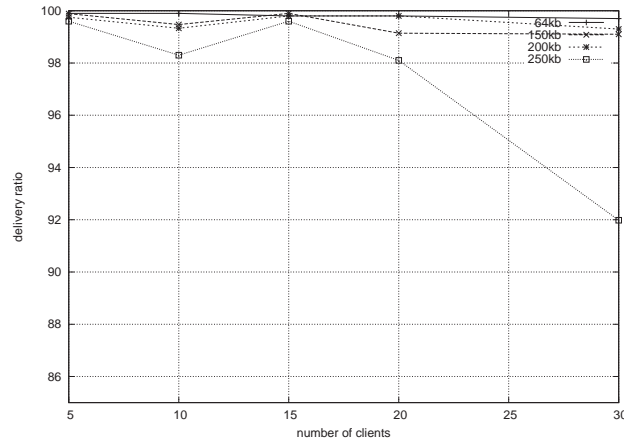


Figure 4.10: Delivery ratio vs group size for different source rates.

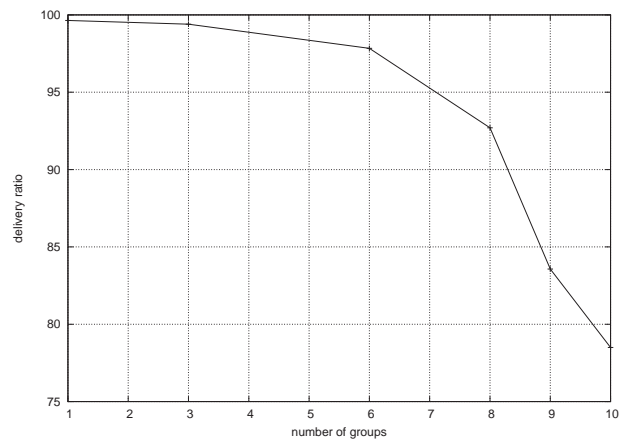


Figure 4.11: Delivery ratio (%) vs number of groups in the network.

Multicast Performance with Mobility

To evaluate the protocol performance with mobility, we consider a scenario consisting of a randomly generated topology of 200 wireless nodes in an area of $1850m \times 1850m$, and one multicast group in which an arbitrary source node sends a CBR traffic of 64kbps for 300 seconds. We run simulations in which we vary the group members, the number of clients (from 5 to 20 nodes, not including the source) and the maximum mobility speed (from $1m/s$ to $10m/s$). The mobility model is the Random Way-point model with a pause time of 10 seconds: nodes choose a random point in the network area and move to it with a constant speed chosen at random between $1m/s$ and the maximum defined value; after they reach their destination, they remain idle for a period equal to the pause

interval and then the same procedure is repeated. Moreover, we consider the following OLSR parameters: a Hello interval of 1 second and TC interval of 5 seconds; we note that the performance of MOST with mobility can be improved by using smaller intervals, at the cost of higher OLSR control message overhead. The simulations are again repeated several times with 5 different seeds. We measure the multicast packet delivery ratio and the traffic load caused by duplicate packets, and we depict the obtained results in Figures 4.12 and 4.13.

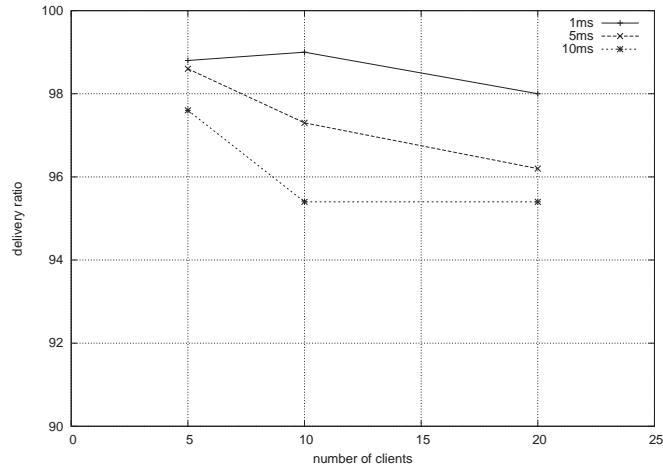


Figure 4.12: Delivery ratio (%) vs group size with different mobility speeds.

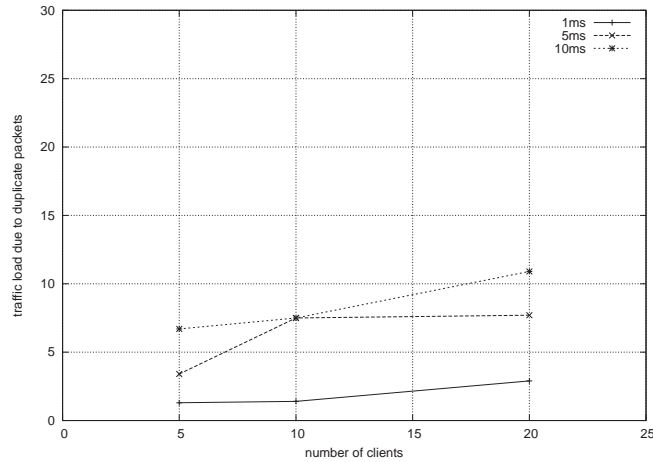


Figure 4.13: Duplicate traffic load (%) vs group size with different mobility speeds.

As we can see, by varying the speed from $1m/s$, $5m/s$ and up to $10m/s$, the delivery ratio remains acceptable, *i.e.*, higher than 95 % for all groups of up to 20 clients. On the other hand, traffic load due to duplicate packets is higher when the mobility speed

or the group size increase. This can be explained by the fact that any change in the topology due to mobility can affect the shared tree. In fact, each client is aware of these changes since it uses the OLSR protocol, and when it recalculates the overlay tree it enters a transition period, during which old and new overlay tree neighbors are maintained. A compromise can be found between the overhead due to duplicate packets and packet delivery ratio by setting a suitable transition period length. We notice that the duplicate traffic load in the network remains small compared to the total traffic load (10% in the worst case), hence the important advantage of improving the packet delivery ratio comes only with a modest performance cost.

4.3 Conclusion

In this chapter, we established performance estimates for multicast routing versus unicast in massively dense ad hoc networks. We showed that multicasting can reduce the overall network load by a factor $O(\sqrt{n})$, for n multicast group members. Consequently, the total capacity of the network for data delivery is significantly increased. Although we used geometric arguments to justify this behavior analytically, we also proposed a protocol which uses only the information provided by a link state routing protocol. The Multicast Overlay Spanning Tree routing protocol was implemented as an extension to OLSR. The protocol is fully distributed, in the sense that each group member computes and maintains the shared multicast tree independently. Being based on overlay trees, it guarantees robustness, while it achieves good performance since OLSR provides topology information allowing to construct an optimal multicast overlay tree. Finally, MOST was tested via ns-2 simulations, and it was shown to achieve the theoretical performance estimates. MOST has also been tested in real network environments, hence we intend to provide measurement studies of the protocol performance in future work.

Chapter 5

Analysis of Traffic Autocorrelations in Large Networks

For the performance evaluation of network algorithms and protocols, data packet arrivals are usually modelled as Poisson processes, as was also the case in the previous chapters. This classic model is attractive due to its unique properties that make the analysis tractable. Moreover, its use can be justified intuitively by the law of large numbers and it has been applied successfully for years in telephone networks. However, recent studies [90] show that it is not sufficient for the characterization of data traffic. In fact, traffic in the Internet displays in many cases long term dependencies, which contradict the memory-less Poisson property. A possible cause of this discrepancy may come from the protocols that are in use in the Internet. In this chapter, we investigate the impact of different widely used protocols, namely of layer 2 protocols such as *Ethernet* [84] or IEEE 802.11 [14], and mainly of the layer 4 TCP protocol (*cf.* [61]). We are able to gain more insight by extending the asymptotic analysis of IEEE 802.11 in Chapter 2 and, for the case of TCP, considering massive network topologies and a large number of concurrent traffics.

By definition, a stationary process has long term dependencies when its autocorrelation function is non-summable. More precisely, let $I(t)$ be the intensity of traffic at time t , and $C(x)$ the covariance of $I(t)$ and $I(t+x)$ when t varies. Function $C(x)$ corresponds to the autocovariance of the traffic intensity and does not depend on t because the process is stationary:

$$C(x) = E[I(t)I(t+x)] - (E[I(t)])^2.$$

We will say that the traffic has long term dependencies when $C(x) \sim Bx^{-\beta}$, with $0 < \beta < 1$.

One of the many undesirable effects of long term dependencies stands in the loss rates in the buffers. Long term dependent traffics generate loss rates that are inverse power functions of the buffer size, contrary to the exponential function of the buffer size

in Poisson models. Consequently, increasing the buffer sizes is prohibitively expensive to significantly reduce the packet loss rate.

The rest of this chapter is organized as follows. In Section 5.1, we utilize the delay analysis from Chapter 2 to derive analytical results about the traffic characteristics. The similarities between IEEE 802.11 and Ethernet permit to apply the analysis in wired scenarios too. We also discuss some implications on TCP performance. In Section 5.2, we aim to characterize the autocorrelations in TCP traffic. We present a model for multi-user TCP and we summarize the analytic derivation of TCP performance. We then use the fact that, according to the previous analysis, the remaining space in the buffer tends to be exponentially distributed in order to derive a simplified model of a single TCP connection with fixed packet loss probability. Based on this model, we characterize the autocorrelation function of a single TCP connection via an estimate of the second eigenvalue of the Markov transition matrix. We investigate the autocorrelation function of several independent TCP connections with heavy tailed round-trip delays. We argue that heavy tailed round trip delays result in heavy tailed traffic autocorrelations and we present a comparison between analytic and simulated results. We already saw such a heavy tailed delay distribution in a wireless setting in Chapter 2. Moreover, measures performed in the Internet in [29] also point to a heavy-tailed round-trip delay distribution.

5.1 Autocorrelations Due to MAC Protocols

In this section, we investigate the impact of MAC layer channel access mechanisms on the traffic characteristics. For this purpose we will consider a model similar to the one introduced in Chapter 2 for wireless nodes. Therefore, we consider again a buffer filled by incoming messages, with a single server that performs the multiple access protocol. Here, we also consider that the buffer is saturated by incoming data, *i.e.*, it always has a packet to transmit to the shared channel. We depict the model in Figure 5.1. The MAC protocol in use can be either IEEE 802.11 or Ethernet (IEEE 802.3). We will calculate the autocorrelation of the traffic actually transmitted by the buffer's server. Therefore, we do not take into account traffic which is subject to collisions or failed transmissions.

In Chapter 2, we showed that the service times in such a buffer are distributed according to a power law, with its shape being determined by the packet collision probability. The mechanism which is essentially the cause for this distribution is the binary exponential back-off procedure, which is in use in IEEE 802.11 to permit the simultaneous use of the channel by multiple sources. This mechanism is also present in Ethernet, hence our analysis will be generalized to that case too. In fact, as we mentioned before, the IEEE 802.11 protocol was conceived as a wireless Ethernet. 802.3 and 802.11 converge on the same logical link control sublayer (defined in 802.2), so they have the same interface to the network layer. Moreover, the two protocols share some similarities in the MAC sublayer as well.

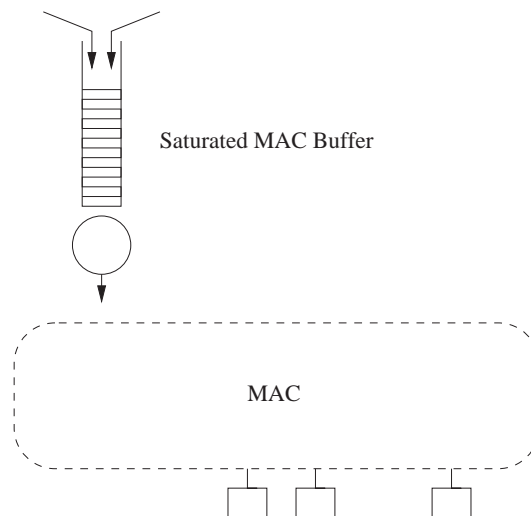


Figure 5.1: Buffer filled with incoming traffic accessing a MAC channel.

Ethernet uses the Carrier Sense Multiple Access with Collision Detection (CSMA/CD) mechanism. A station only transmits when the medium is idle and if a collision is detected the transmission is interrupted. Collisions can be detected by looking at the power or pulse width of the received signal and comparing it to the transmitted signal. Hence, a station must listen to the cable while it is transmitting. When two stations transmit simultaneously, they will both detect the collision almost immediately and terminate the transmission of the already damaged frames, hence saving time and bandwidth (this is not the case in IEEE 802.11, where collision detection could not be used because of the hidden and exposed terminal problems). After a collision the stations use the binary exponential back-off mechanism which we described in Section 2.1 (referring to 802.11) to randomize their access to the channel and avoid further collisions. In Ethernet the minimum contention window size is 1 slot, while the maximum is 1024 slots.

Therefore, the service delay analysis from Section 2.2.1 can also be adapted to this case, to show that the complementary cumulative distribution of service times decays as a power law with exponent $B > 0$. If we use the same notation for the service delay probability generating function $\beta(z)$, we have the expansion of the complex function $\beta(z)$ around $z = 1$ of the form (*cf.* Theorem 1 in Chapter 2):

$$\beta(z) = \begin{cases} 1 + (1-z)v(1-z) + C(1-z)^B + O((1-z)^{B+1}), & B \notin \mathbb{Z} \\ 1 + (1-z)v(1-z) + C(1-z)^B \log \frac{1}{1-z} + O((1-z)^{B+1}), & B \in \mathbb{Z} \end{cases} \quad (5.1)$$

where $v(1-z)$ is an analytic function and C is a constant.

The constant B depends on the collision probability of packets when trying to access the channel. Since, the operation of Ethernet is similar to 802.11 with the addition of

collision detection, but the back-off mechanism is identical, we use the same formula as in Section 2.2.1, although a slight modification must be made in the analysis to account for the premature ending of a packet transmission when a collision is detected¹. However, the result will not differ qualitatively. In both protocol cases, we have $B = -\log_2 p$, where p is the packet collision probability. Moreover, we take here into consideration the case of integer B .

We note that instead of the total amount of traffic going through the buffer, we can also model the inter-arrival process of an individual connection, when a protocol using an acknowledgement mechanism is used (such as TCP). Hence, we comment briefly on this case too whenever possible. In this scenario, before each transmission of a new packet the sender must wait for an acknowledgement receipt.² Therefore, instead of considering the service time to evaluate packet inter-arrivals, we must use an estimate for the delay of the packet to reach its destination plus the acknowledgment delay. This can be accounted for since we have analyzed the packet delays including queuing also. We can adjust our study to this scenario by setting $B = 1 - \log_2 p$. In a wireless multi-hop setting, p is the highest collision probability experienced in the packet's route (*cf.* Section 2.2.1).

5.1.1 Throughput Analysis

At first, we calculate the expected cumulative throughput in the described scenario. We denote $h(n)$ the cumulative throughput at time n , *i.e.*, the total number of packets sent after n time units.

Let S_i be the service time experienced by packet i . We have the probability that the number of packets sent is larger than a number k :

$$P(h(n) > k) = P(S_1 + S_2 + \dots + S_{k+1} \leq n). \quad (5.2)$$

In order to compute this probability, we define the bivariate generating function $H(z, u) = \sum_{k,n} P(h(n) > k) u^k z^n$, where z marks the time and u marks the number of packets sent. If all service times are independent and distributed according to the generating function $\beta(z) = \sum_n P(S = n) z^n$, we have:

$$\begin{aligned} H(z, u) &= \sum_k u^k \sum_n P(S_1 + S_2 + \dots + S_{k+1} \leq n) z^n \\ &= \sum_k u^k (\beta(z))^{k+1} \frac{1}{1-z} \\ &= \frac{\beta(z)}{1-z} \frac{1}{1-u\beta(z)}. \end{aligned} \quad (5.3)$$

¹In case of Ethernet, the generating function z^L in (2.2), corresponding to the packet transmission length, can be substituted with an analytic function, namely with a polynomial $q(z)$ of degree at most L .

²We can assume here for simplicity that one packet is sent at a time. The TCP protocol will be described in more detail in a following section, where we will also model its sliding window mechanism.

We can compute the average cumulative throughput at time n as follows:

$$\begin{aligned}\mathbf{E}[h(n)] &= \sum_k P(h(n) > k) \\ &= [z^n]H(z, 1) \\ &= [z^n] \frac{\beta(z)}{1-z} \frac{1}{1-\beta(z)},\end{aligned}\tag{5.4}$$

where $[z^n] f(z)$ stands for the coefficient of z^n in the generating function $f(z)$.

Based on (5.1), we now consider the expansion of $H(z, 1)$ around $z = 1$, to deduce the asymptotic behavior of $\mathbf{E}(h(n))$ when $n \rightarrow \infty$. Clearly, we have

$$H(z, 1) = \frac{1}{1-z} \frac{1}{1-\beta(z)} (1 + O(1-z)).\tag{5.5}$$

We consider three different cases according to the value of B .

1. $B > 1$.

To simplify the computation, we assume first that $B \notin \mathbb{Z}$:

$$\begin{aligned}H(z, 1) &= \frac{1}{-(1-z)^2 v(0) - C(1-z)^{B+1} + O((1-z)^{B+2})} (1 + O(1-z)) \\ &= \frac{1}{-(1-z)^2 v(0) \left(1 + \frac{C}{v(0)} (1-z)^{B-1} + O((1-z)^B)\right)} (1 + O(1-z)),\end{aligned}$$

and using the fact that $\frac{1}{1-x} = 1 + x + O(x^2)$, when $x \rightarrow 0$:

$$\begin{aligned}H(z, 1) &= \frac{1}{-(1-z)^2 v(0)} \left(1 - \frac{C}{v(0)} (1-z)^{B-1} + O((1-z)^B)\right) \\ &= \frac{1}{-(1-z)^2 v(0)} + \frac{C}{v(0)^2} (1-z)^{B-3} + O((1-z)^{B-2}).\end{aligned}$$

Therefore, with Flajolet-Odlyzko singularity analysis we can deduce the behavior of $\mathbf{E}(h(n))$ when $n \rightarrow \infty$:

$$\begin{aligned}\mathbf{E}(h(n)) &= \frac{n}{-v(0)} + \frac{C}{v(0)^2 \Gamma(3-B)} n^{2-B} + O(n^{1-B}) \\ &= \frac{n}{-v(0)} (1 + o(1)).\end{aligned}\tag{5.6}$$

Note that $-v(0)$ corresponds to the mean value of the service time. In case $B \in \mathbb{Z}$, the final asymptotic result remains unchanged.

2. $B = 1$.

$$\begin{aligned} H(z, 1) &= \frac{1}{-C(1-z)^2 \log \frac{1}{1-z} + O((1-z)^2)} (1 + O(1-z)) \\ &= \frac{1}{-C(1-z)^2 \log \frac{1}{1-z}} \left(1 + O(\log^{-1} \frac{1}{1-z}) \right) \\ &= \frac{1}{-C(1-z)^2 \log \frac{1}{1-z}} + O\left((1-z)^{-2} \log^{-2} \frac{1}{1-z} \right). \end{aligned}$$

Therefore (cf. [51], Chapter VI),

$$\begin{aligned} \mathbf{E}[h(n)] &= \frac{n}{-C \log n} \sum_{j \geq 1} \frac{C_j^*}{(\log n)^j} + O(n \log^{-3} n) \\ &= \frac{(1-\gamma)n}{C \log^2 n} + O(n \log^{-3} n) \\ &= \frac{(1-\gamma)}{C} \frac{n}{\log^2 n} (1 + o(1)), \end{aligned} \tag{5.7}$$

where $C_k^* = \left. \frac{d^k}{ds^k} \frac{1}{\Gamma(s)} \right|_{s=2}$, and γ is Euler's constant.

3. $B < 1$

$$\begin{aligned} H(z, 1) &= \frac{1}{-C(1-z)^{B+1} + O((1-z)^2)} (1 + O(1-z)) \\ &= \frac{1}{-C(1-z)^{B+1}} + O((1-z)^{-2B}). \end{aligned}$$

Therefore,

$$\mathbf{E}[h(n)] = \frac{n^B}{-\Gamma(B+1)C} + O(n^{2B-1}) = \frac{n^B}{-\Gamma(B+1)C} (1 + o(1)). \tag{5.8}$$

This analysis verifies the expected result that the throughput is inversely proportional to the average service time, when the latter is finite. Interestingly enough, we can also compute the cumulative throughput behavior even in cases where the average service time tends to infinity (when $B \leq 1$). As a side note, we mention that this case refers to collision probabilities of at least 25% for the saturated buffer, and 12.5% if we consider the individual throughput of a TCP connection. In practice the average delays will not be infinite because of the limit in the number of retransmissions. However, the throughput will still suffer a significant decrease in these cases.

In Figures 5.2, 5.3 and 5.4, we verify the theoretical results with measurements which are obtained by simulating the packet inter-arrival process with Matlab. We present one simulation for each of the three cases concerning the possible values of B . In the graphs, we can clearly notice that, as the value of B decreases, there are increasingly longer periods during which no packets are transmitted, hence resulting in the sub-linear growth of the cumulative throughput.

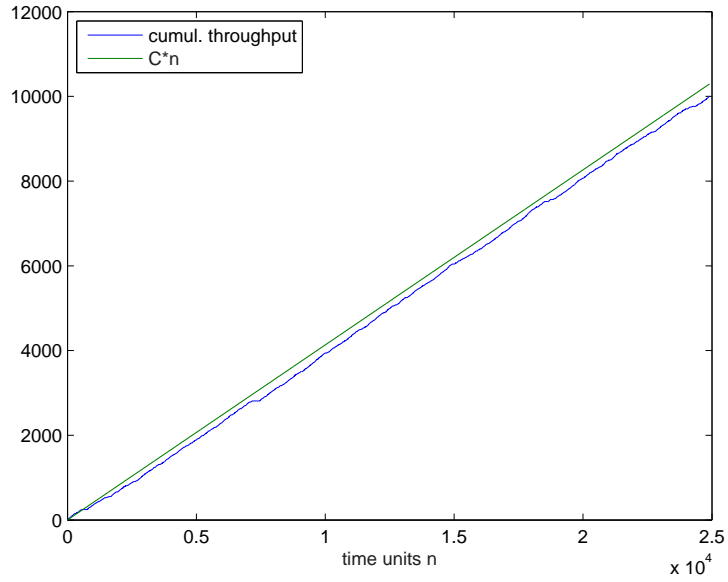


Figure 5.2: Linear cumulative throughput evolution versus time n when $B = 1.5$.

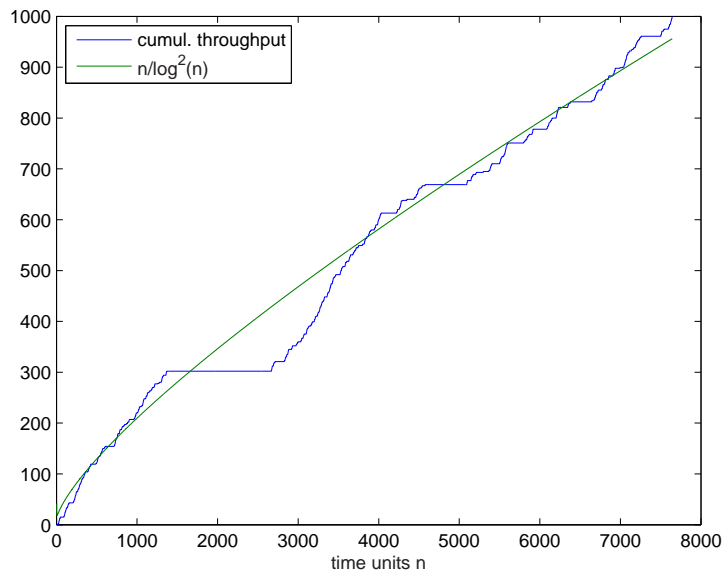


Figure 5.3: Cumulative throughput in $\frac{n}{\log^2 n}$ versus time n in the limit case of $B = 1$.

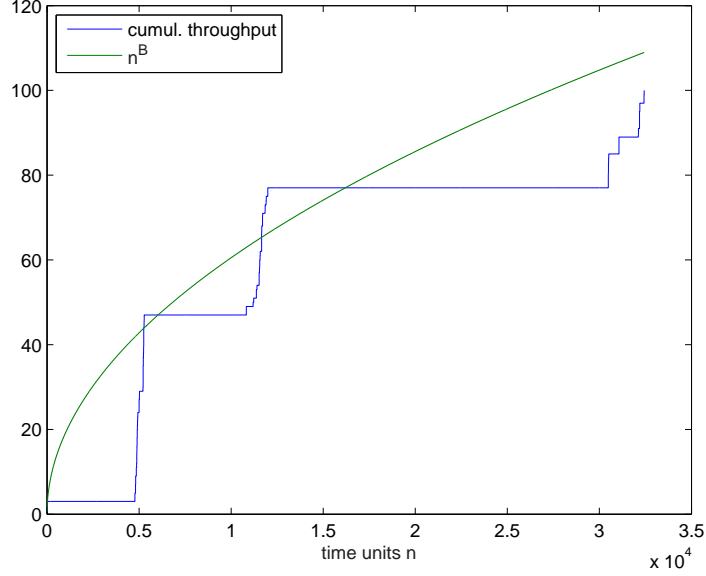


Figure 5.4: Cumulative throughput in n^B versus time n when $B = 0.5$.

5.1.2 Autocorrelation Analysis

Based on the above, we can now evaluate the autocorrelation of the traffic. Let $I(t)$ be the traffic intensity at time t , *i.e.*, the number of packets emitted at time t . Since we consider a single source $I(t)$ can be either 0 or 1. The autocovariance function of the traffic intensity is:

$$\begin{aligned}
 C(x) &= \mathbf{E}[I(t)I(t+x)] - \mathbf{E}[I(t)]^2 \\
 &= \mathbf{E}[I(t+x)|I(t)=1] \cdot P(I(t)=1) - \mathbf{E}[I(t)]^2 \\
 &= \sum_t 1P(I(t+x)=1|I(t)=1) \cdot P(I(t)=1) - \mathbf{E}[I(t)]^2 \\
 &= \mathbf{E}[I(t)] \cdot (P(I(t+x)=1|I(t)=1) - \mathbf{E}[I(t)]). \tag{5.9}
 \end{aligned}$$

From the throughput analysis it comes that, in stationary state, $\mathbf{E}[I(t)] = \frac{1}{-v(0)}$, when $B > 1$, while $\mathbf{E}[I(t)] = 0$, when $B \leq 1$. Therefore the autocovariance is also $C(x) = 0$ when $B \leq 1$, although the convergence is very slow, hence we will discuss this case too.

We have:

$$P(I(t+x)=1|I(t)=1) = \sum_{k \geq 1} P(S_1 + S_2 + \dots + S_k = x),$$

where S_i is the service time of packet i emitted after time t .

We define again a bivariate generating function

$$i(z, u) = \sum_{k,x} P(I(t+x) = 1 | I(t) = 1) u^k z^x, \quad (5.10)$$

where z marks the time and u marks the number of packets sent. Based on the previous assumptions on service delays:

$$\begin{aligned} i(z, u) &= \sum_{k \geq 1} u^k \sum_x P(S_1 + S_2 + \dots + S_k = x) z^x \\ &= \sum_{k \geq 1} u^k (\beta(z))^k \\ &= \frac{\beta(z)}{1 - u\beta(z)} = (z-1)H(z, u). \end{aligned} \quad (5.11)$$

We have:

$$P(I(t+x) = 1 | I(t) = 1) = [z^x] i(z, 1) = [z^x] (z-1)H(z, 1).$$

Hence we obtain the asymptotic behavior using the previous results and the expansion of $i(z, 1)$ around $z = 1$.

We consider again the different possible values of B .

1. $B > 1$.

Again, to alleviate the analysis, we first assume that $B \notin \mathbb{Z}$.

$$i(z, 1) = \frac{1}{-(1-z)v(0)} + \frac{C}{v(0)^2} (1-z)^{B-2} + O((1-z)^{B-1}).$$

When $x \rightarrow \infty$,

$$P(I(t+x) = 1 | I(t) = 1) = \frac{1}{-v(0)} + \frac{C}{v(0)^2 \Gamma(2-B)} x^{1-B} + O(x^{-B}).$$

Therefore the autocovariance decreases according to the following:

$$C(x) = \frac{C}{-v(0)^3 \Gamma(2-B)} x^{1-B} + O(x^{-B}), \quad (5.12)$$

and the traffic has long term dependencies when $1 < B \leq 2$, *i.e.*, when the variance of the service times is infinite. In case $B \in \mathbb{Z}$, in the asymptotic estimate there will be only a slight change in the constants, which will not involve the Gamma function.

2. $B = 1$.

$$i(z, 1) = \frac{1}{-C(1-z) \log \frac{1}{1-z}} + O((1-z)^{-1} \log^{-2} \frac{1}{1-z}),$$

and

$$P(I(t+x) = 1 | I(t) = 1) = \frac{(1-\gamma)}{C} \frac{1}{\log^2 x} (1 + o(1)). \quad (5.13)$$

3. $B < 1$.

$$i(z, 1) = \frac{1}{-C(1-z)^B} + O((1-z)^{-2B-1}).$$

Therefore,

$$P(I(t+x) = 1 | I(t) = 1) = \frac{x^{B-1}}{-\Gamma(B)C}(1 + o(1)). \quad (5.14)$$

In the two latter cases, the obtained result corresponds to the asymptotic behavior of the traffic autocorrelation at time t , provided that there was a packet transmission. Interestingly enough, the decrease is very slow in the limit case when $B = 1$, due to the fact that the throughput decreases also at a very slow rate.

We note here the undesirable effect on TCP throughput when the delay variance tends to become large, because of the way the protocol detects lost packets. The detection occurs after a time-out according to a round trip delay estimate. Even though this time-out accounts for the delay variance, it cannot give reliable results when the variance tends to infinity. From the analysis, it comes that this can happen for collision rates as low as 6.25% in multi-hop wireless environments. In other words, TCP intensifies the effects of the delay variations, which we identified in Chapter 2. Under this simple model, we can think of an optimization for TCP performance which consists in excluding the links with high collision probabilities, as it was done in Chapter 3 using link hysteresis strategies.

We verify the theoretical results using Matlab simulations, as in the previous section. Instead of measuring directly the autocorrelation traffic of the packet arrival process, we consider the aggregated variance, which is a quantity that is used to determine the stronger property of whether a process is self-similar in [78]. Let X be the stationary stochastic process representing the number of packets that are sent in a time unit. Let X^m , for $m = 1, 2, 3 \dots$, be the stationary process obtained by averaging the original series X over non-overlapping blocks of size m . This corresponds to counting the number of packets sent in increasing intervals of size m . After measuring X^m for a large range of intervals m , we compute the aggregated variance $\text{var}(X^m)$. In the case of long term dependent processes, the decrease rate of the aggregated variance $\text{var}(X^m)$ versus m corresponds to the decrease rate of the auto-correlation function (*i.e.*, $\text{var}(X^m) \sim m^{-b}$, with $0 < b < 1$). On the other hand, when the autocorrelation function decreases more rapidly, the aggregated variance decreases like the reciprocal of the sample mean (*i.e.*, $\text{var}(X^m) \sim m^{-1}$). These properties permit us to compare the measurements with the theoretical bounds. Therefore, we present in Figures 5.5, 5.6, 5.7 the curves that we obtained from simulations for different values of B , following the three possible ranges of the analysis, all resulting in long term dependencies.

We note that the theoretical analysis verifies the experimental evidence concerning Ethernet traffic in [78], and justifies the ON/OFF source model also introduced in the same paper, since we make a link between the exponential back-off mechanism in Ethernet and long term dependencies (see also [53] for more experimental evidence demonstrating the impact of the exponential back-off in traffic autocorrelations).

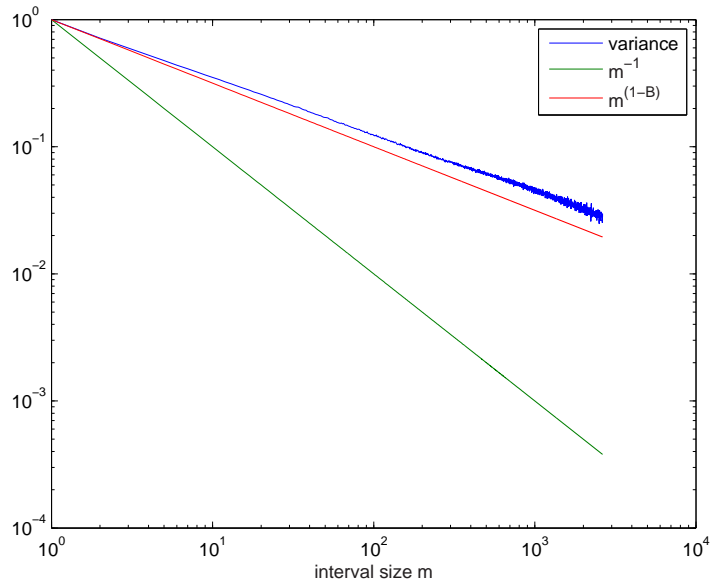


Figure 5.5: Aggregated variance in m^{1-B} versus measure interval m , when $B = 1.5$.

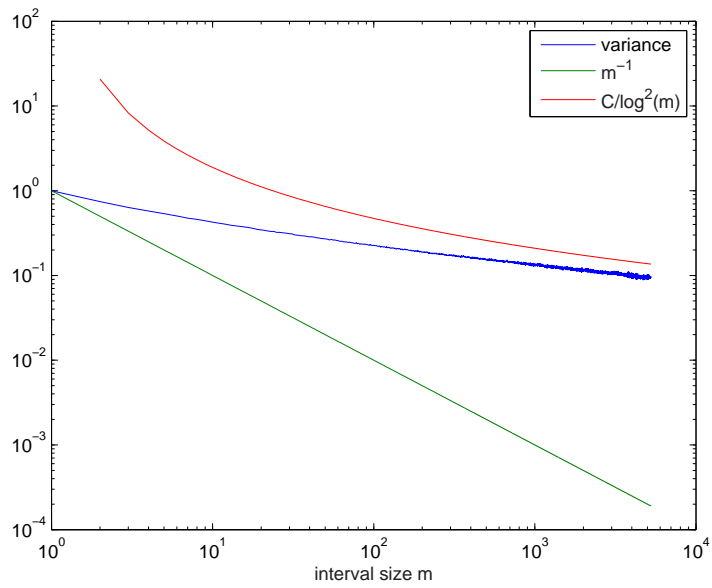


Figure 5.6: Aggregated variance in $\frac{1}{\log^2 m}$ versus measure interval m , in the limit case of $B = 1$.

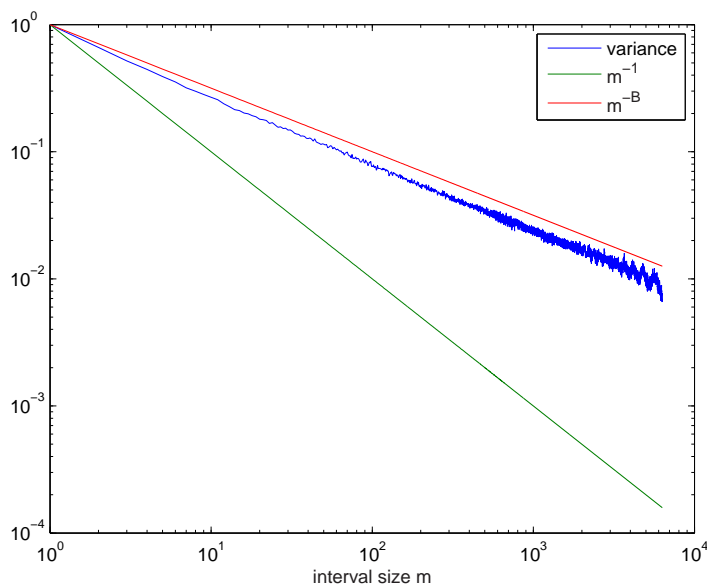


Figure 5.7: Aggregated variance in m^{-B} versus measure interval m , when $B = 0.5$.

5.2 Autocorrelations in TCP Traffic

In this section, we analyze the impact of higher layer protocols, namely TCP, in the traffic autocorrelations. This impact of the higher layer protocols is expected to be more noticeable in larger time scales. On the other hand, in smaller time scales, it is the lower protocols that play the most important role. We will focus here on the larger time scale and separate the analysis from the previous section, in order to be able to characterize the autocorrelation function of TCP traffic in a more general setting.

5.2.1 TCP Protocol Overview and Models

The dynamic window based protocol TCP [61] is widely used over the Internet, and carries more than 90% of the overall traffic. The reason of the success of TCP is mainly based on its highly dynamic nature, that makes it able to adapt itself to any kind of network capacity, from a few bits to several gigabits per second. TCP was formally defined in RFC 793 [44]. Some clarifications and bug fixes are detailed in RFC 1122 [45]. Extensions are given in RFC 1323 [62].

In TCP, packets are transmitted in sequence and must be acknowledged by the end-user. A packet is considered to be lost when the acknowledgement does not arrive within the estimated round trip delay, or when duplicate acknowledgements for the same packet are received (this occurs when packets are lost but subsequent packets with

larger sequence numbers are transmitted successfully). A packet loss is considered as a congestion event.

In order to cope with the round trip delay, several packets are transmitted in advance without waiting for acknowledgement. The set of unacknowledged packets is called the window and its size varies in order to handle congestion.

- (i) when no packet is lost in a window (successful window), the next window size is incremented by 1 unit: $W \leftarrow W + 1$.
- (ii) when a packet loss occurs (failed window), packet retransmission starts from this packet, but with a window size halved: $W \leftarrow \lfloor W/2 \rfloor$.

Our description constitutes a simplification of the complex operation of the TCP protocol, but it provides a sufficiently accurate abstraction of the algorithms used in TCP today. The complete reference of TCP operations can be found in the RFCs (for instance RFC 2581 [20], RFC 2582 [52] in addition to the RFCs cited previously). The protocol also includes more mechanisms to make the operation more efficient, such as self-clocking and slow start. The latter feature makes TCP more reactive in network condition changes.

Self-clocking The server synchronizes the transmission of its new packets with the acknowledgement returns, leading to a *self-clocking* of packet transmissions. Packets can be acknowledged in batch.

Slow start In the *slow start* phase the window size doubles at each full successful window, until it reaches a predefined threshold. TCP enters the slow start phase after a packet transmission time-out.

Multi-user TCP

In this section, we describe a model for multiple TCP flows which constitutes the initial basis for our analysis. A detailed description and analysis of this model can be found in [19, 7]. However, here we give only the most relevant results.

We consider a large number N of parallel TCP connections towards a bottleneck router with a finite buffer of capacity B connected to a slow network interface with service rate μ (see Figure 5.8). In particular, the network is divided into a local loop with relatively low speed and a backbone with high throughput, hence we assume that all congestion occurs in the bottleneck buffer.

We assume that the transmitted files are infinite. We also assume that all packets in a TCP window arrive in a single batch. The round trip time (RTT) has a fixed component

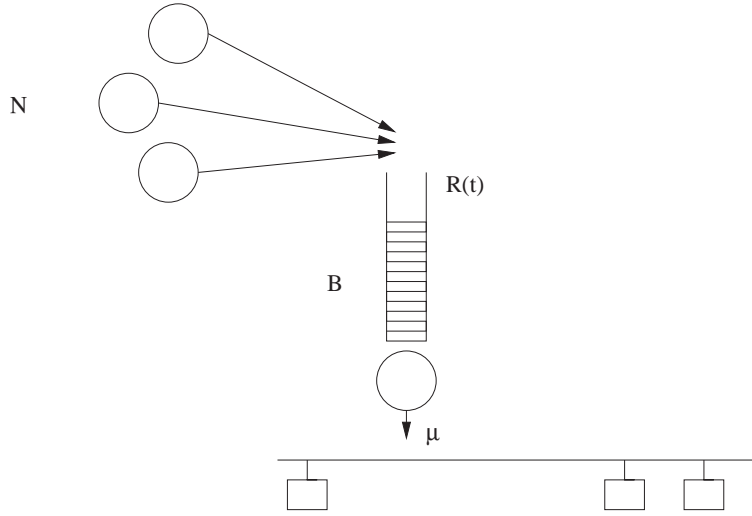


Figure 5.8: N TCP connections towards the same bottleneck buffer.

$\frac{B}{\mu}$ (the maximum sojourn time in the buffer) and a random exponential delay of mean $\frac{N}{\lambda}$, assumed to be much smaller than $\frac{B}{\mu}$.

In such a model the buffer is expected to be full most of the time. Indeed, if we call $R(t)$ the available room in the buffer at time t , we have $R(t) = O(1)$ when $N \rightarrow \infty$. This result is derived from the analysis in [19, 7], where it is shown that, when the system is stationary, the probability that the available room on top of the buffer $R(t)$ exceeds x has the expression:

$$P(R(t) \geq x) \rightarrow \exp(-ax),$$

for some constant $a > 0$, which corresponds to the probability of the buffer being full.

This observation is used in the following section to derive a simplified model where there is a fixed packet loss probability a and moreover the losses are independent.

We denote $w(y)$ the density function of the TCP window size distribution when the system is in steady state. When $a \rightarrow 0$ and the round trip delay is large, we have $w(y) = \sqrt{a}g(\sqrt{a}y) + O(a)$, where $g(y)$ satisfies the differential equation:

$$yg(y) + g'(y) = 4yg(2y).$$

This equation can be explicitly solved and the solution is depicted in Figure 5.9. In passing, note that the TCP window sizes vary greatly although all connections share the same bottleneck buffer. In this case, we have the asymptotic estimate for parameter a :

$$\sqrt{a} = (1 + O(\sqrt{a})) \frac{g^*(2)}{\frac{B}{N} + \frac{\mu}{\lambda}},$$

with $g^*(2) = \int_0^\infty yg(y)dy \simeq 1.3098$.

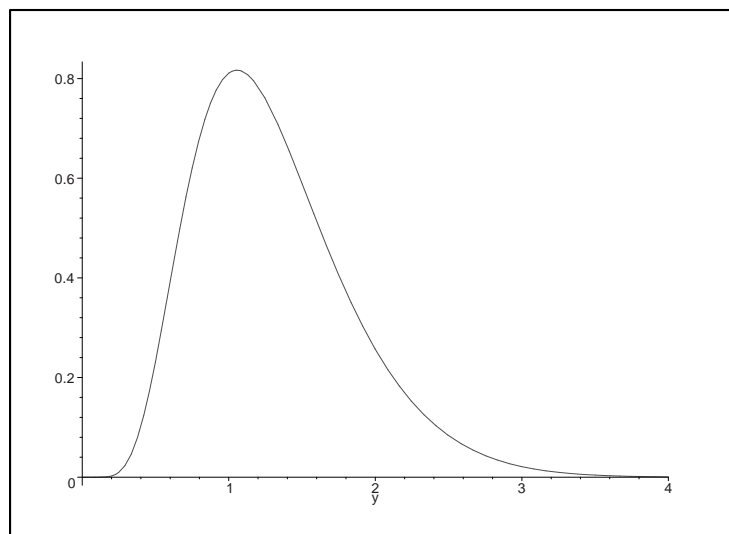


Figure 5.9: Limiting function $g(x)$ of the window size distribution.

Fixed Loss Probability Model

We now derive a simplified model of a single TCP connection, in order to analyze the autocorrelation function of TCP traffic. We assume that the packet loss probability is constant and equal to a , and that the losses are independent. This is justified by the previous model where a is constant. For simplicity, we turn to a discrete time model, where the unit is the RTT (we assume that we collect all buffer changes within one RTT), which does not vary much in the considered time scale since it is composed of a large fixed delay and a much smaller random processing time³. Hence, we can model the window size adaptation with a Markov chain. A similar model has been studied via simulation in [48].

We study the behavior of a TCP connection transferring an infinite file, during the congestion avoidance phase⁴. If there is a packet loss in a window transmission, the retransmission of packets resumes from the first lost packet but with a window size divided by two: $W = \lfloor W/2 \rfloor$, otherwise the window size increases by one: $W = W + 1$. We note that whether there are more losses or not after the first loss in a window is of no importance in the validity of our model.

³We note that in contrast to the results of the previous section, we take here the weakest assumption with regards to traffic autocorrelations. The reason is to model a more general context, and to show in the following section that, even in this case, TCP traffic can generate long term dependencies.

⁴Although most of Internet connections are short-lived, the majority of traffic is carried by persistent connections, which correspond to this case.

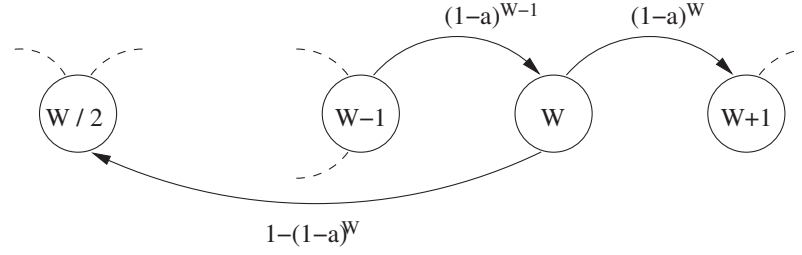


Figure 5.10: Markovian TCP model.

We deduce the following probabilities for $a \ll 1$:

$$Pr(\text{no loss in a window}) = (1-a)^W \approx e^{-aW} \quad (5.15)$$

$$Pr(\text{loss in a window}) = 1 - (1-a)^W \approx 1 - e^{-aW} \quad (5.16)$$

We now define the TCP discrete time Markov chain. The states are the window sizes and the transition probabilities can be calculated from (5.15) and (5.16). The resulting Markov chain is depicted in Figure 5.10. In reality, the receiver announces a maximum window size, which means that the Markov chain is finite. As the tendency in the Internet is towards increasing this value, we will ignore it in the analysis. We suppose that it corresponds to a very large window size, which is never reached in practice.

We denote π_i the stationary probability of state i . In steady state, the following equations hold when $a \ll 1$:

$$\pi_1 = a\pi_1 + (1 - e^{-2a})\pi_2 + (1 - e^{-3a})\pi_3 \quad (5.17)$$

$$\pi_k = e^{-a(k-1)}\pi_{k-1} + (1 - e^{-a(2k+1)})\pi_{2k+1} + (1 - e^{-a2k})\pi_{2k}, k \geq 2 \quad (5.18)$$

Let \mathbf{P} be the transition matrix of the TCP Markov chain. It is easy to see that \mathbf{P} is stochastic, irreducible and aperiodic. Thus, according to the Perron-Frobenius theorem, its dominant eigenvalue is 1 with corresponding right eigenvector $\boldsymbol{\pi}$, which gives the stationary distribution of the Markov chain. We can also deduce that $\lim_{n \rightarrow \infty} \mathbf{P}^n = \boldsymbol{\pi} \cdot \mathbf{1}$, where $\mathbf{1}$ is a line vector of ones.

The matrix \mathbf{P} is of the following form:

$$\mathbf{P} = \begin{pmatrix} 1 - e^{-a} & 1 - e^{-2a} & 1 - e^{-3a} & 0 & \dots \\ e^{-a} & 0 & 0 & 1 - e^{-4a} & \\ 0 & e^{-2a} & 0 & 0 & \\ & 0 & e^{-3a} & 0 & \\ & & 0 & e^{-4a} & \dots \\ \vdots & & & \vdots & \ddots \end{pmatrix} \quad (5.19)$$

If the initial distribution of windows is $\boldsymbol{\pi}(0)$, then after n RTT's: $\boldsymbol{\pi}(n) = \mathbf{P}^n \boldsymbol{\pi}(0)$. From the spectral decomposition of the matrix, we get that the convergence rate is exponential:

$$\|\boldsymbol{\pi}(n) - \boldsymbol{\pi}\| = O(\rho^n), \quad (5.20)$$

for every ρ such that $|\lambda_2| < \rho < 1$, where λ_2 is the second largest eigenvalue of \mathbf{P} .

5.2.2 Autocorrelation Function of a Single TCP Connection

Analysis under the Fixed Loss Probability Model

The traffic intensity $I(t)$ at time t is a function of the present window size $W(t)$ and the round trip time, which we denote as D :

$$I(t) = \frac{W(t)}{D} \text{ packets/second.} \quad (5.21)$$

We calculate the traffic autocovariance function $C(x)$ at time t :

$$C(x) = \text{cov}[I(t), I(t+x)] = E[I(t)I(t+x)] - (E[I(t)])^2. \quad (5.22)$$

As we are dealing with a stationary process, $C(x)$ does not depend on time t , so we can fix $t = 0$. If we express the time in RTT multiples n , the first term of (5.22) becomes:

$$E[I(t)I(t+nD)] = E[I(0)I(nD)] = \frac{1}{D^2} E[W(0)W(nD)]. \quad (5.23)$$

We continue by supposing that the initial window size is k and averaging on all k :

$$\begin{aligned} E[I(0)I(nD)] &= \frac{1}{D^2} E[E(W(nD)k|W(0) = k)] \\ &= \frac{1}{D^2} \sum_k k E(W(nD)|W(0) = k) \boldsymbol{\pi}(0)_k, \end{aligned} \quad (5.24)$$

where $\boldsymbol{\pi}(n)_i$ is the probability of the window being of size i after n RTT's.

We want to calculate the autocovariance of a system that has reached equilibrium, so we can assume that the initial distribution $\boldsymbol{\pi}(0)$ is the stationary distribution $\boldsymbol{\pi}$:

$$\begin{aligned} E[I(0)I(nD)] &= \frac{1}{D^2} \sum_k k E(W(nD)|W(0) = k) \boldsymbol{\pi}_k \\ &= \frac{1}{D^2} \sum_k k \sum_l (\mathbf{P}^n \mathbf{1}_k)_l \boldsymbol{\pi}_k, \end{aligned} \quad (5.25)$$

where $\mathbf{1}_k$ is a column vector with all entries being 0 except for entry k which is 1.

If we define \mathbf{p} as a column vector such that $\mathbf{p}_i = i\pi_i$, and \mathbf{u} as a line vector such that $\mathbf{u}_i = i$, then:

$$E[I(0)I(nD)] = \frac{1}{D^2} \sum_l l (\mathbf{P}^n \mathbf{p})_l = \frac{1}{D^2} \mathbf{u}(\mathbf{P}^n \mathbf{p}). \quad (5.26)$$

For $n \rightarrow \infty$ we have:

$$\begin{aligned} \frac{1}{D^2} \mathbf{u}(\lim_{n \rightarrow \infty} \mathbf{P}^n \mathbf{p}) &= \frac{1}{D^2} \sum_l l \left(\lim_{n \rightarrow \infty} \mathbf{P}^n \mathbf{p} \right)_l = \frac{1}{D^2} \sum_l l (\boldsymbol{\pi} \cdot \mathbf{1} \cdot \mathbf{p})_l \\ &= \frac{1}{D^2} \sum_l l (\pi_l \sum_k k \pi_k) = (E[I(0)])^2. \end{aligned} \quad (5.27)$$

Combining (5.22), (5.26) and (5.27) we get the autocovariance function:

$$\begin{aligned} C(nD) &= E[I(0)I(nD)] - (E[I(0)])^2 \\ &= \frac{1}{D^2} \mathbf{u} \left(\mathbf{P}^n \mathbf{p} - \lim_{n \rightarrow \infty} \mathbf{P}^n \mathbf{p} \right). \end{aligned} \quad (5.28)$$

From the spectral decomposition of \mathbf{P} , as in (5.20), we conclude that the autocovariance function decreases exponentially with rate $O(\rho^n)$ for all ρ such that $|\lambda_2| < \rho < 1$. By normalizing we obtain the autocorrelation function, which is also $O(\rho^n)$. The next step in our analysis is to calculate numerically the eigenvalues of \mathbf{P} for a large scale of error rates. The calculations are performed with Matlab, by fixing a maximal window size of 1000 packets, thus truncating the matrix \mathbf{P} . The probability of reaching the maximum window size is close to 0 for the error rates we consider, so ignoring larger values does not affect significantly our results. In the equivalent continuous model of TCP, we scaled window sizes by a factor $\frac{1}{\sqrt{a}}$ to obtain the limit distribution. We expect to find a similar factor in the autocovariance function. This observation leads us to approximate the spectral gap of the TCP Markov chain by expression $C\sqrt{a}$, where C is a constant. In Figure 5.11, we compare the calculated spectral gap values, for different error rates a , and the values corresponding to the proposed approximation for $C = 1.6$. In all our calculations, the second eigenvalue is real, of multiplicity 1, which means that the autocovariance is $O(\lambda_2^n)$. More precisely, we have the following first order approximation:

$$C(nD) = \frac{A}{D^2} \lambda_2^n \approx \frac{A}{D^2} e^{-C\sqrt{a}n}, \quad (5.29)$$

where A is another constant.

Observe that a very small error rate results in a second eigenvalue close to 1, meaning that the autocorrelation function decreases very slowly (although exponentially).

Simulations of a Single TCP Connection

To verify that our simplified model can predict the autocorrelation of a real TCP connection, we conducted a number of simulations with the network simulator ns-2 [46].

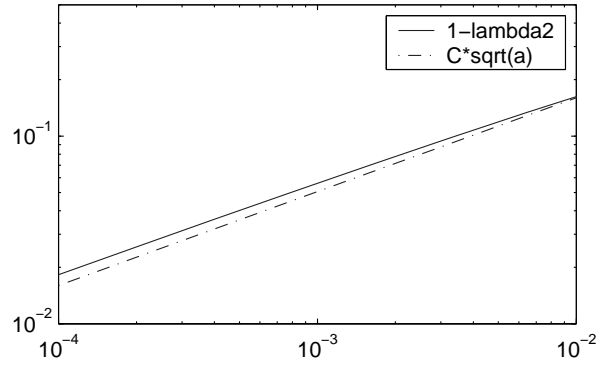


Figure 5.11: Spectral gap of the TCP Markov chain for different error rates a .

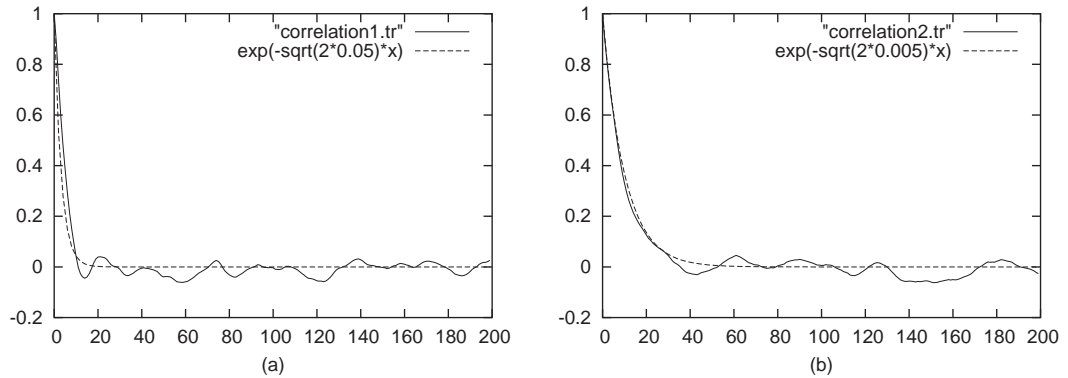


Figure 5.12: TCP traffic autocorrelations for error rates (a) $a = 0.05$, (b) $a = 0.005$. The time unit is the RTT.

In the simulations, the RTT is mainly caused by the link delays and the error rate is constant and due to a loss agent. By measuring the number of packets transmitted during an interval equal to the RTT, we obtain the traffic intensity $I(t)$. We then calculate the covariance of $I(t)$ and $I(t+x)$ when t varies and we normalize to obtain the traffic autocorrelation.

We present the results for two models with different error rates, compared to the calculations of the previous section. The duration of these simulations is 1000 seconds of simulated time, and we start making measurements after waiting for the system to stabilize for 100 seconds. In Figure 5.12 we draw the autocorrelation for error rates $a = 0.05$ and $a = 0.005$. The time unit is the round trip delay, which in this case is equal to 100ms.

The oscillations are due to the finite duration of the simulations.

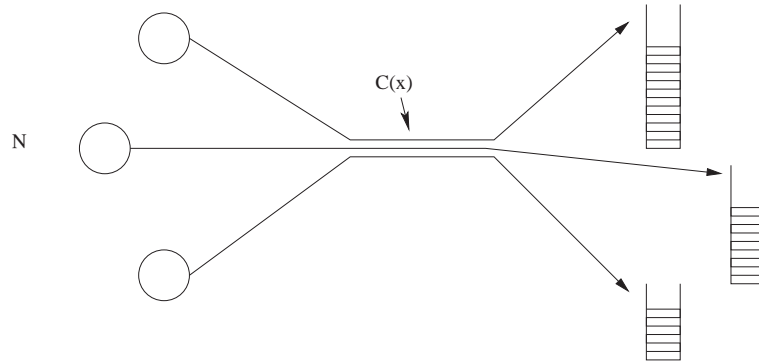


Figure 5.13: Several TCP connections sharing the same link.

5.2.3 Long Term Dependencies in Multi-user TCP

In this section we will show that long term dependencies can arise from heavy tailed round trip delays. In the first subsection we show that a link shared by several TCP connections with round trip times with a heavy tailed distribution generates long term dependence. In the second subsection we provide simulations to compare with the theoretical results.

The remaining question is to figure out whether or not the round trip delays are heavy tailed. Recently it has been discovered that the internet topology contains numerous heavy tailed features. Among them are the router degree, router reachability degree and the length of paths inside the internet [30, 33, 17]. In [29] there is evidence that the RTT distribution is also heavy tailed, *i.e.*, the RTT complementary cumulative density function $P(RTT > x)$ corresponds to a power law with exponent approximately 1.5. Equivalently, the rank of the RTTs follows a power law of exponent $2/3$. Furthermore, in Chapter 2 we showed that the delay distribution in a wireless setting can also be expressed with a power law.

Autocorrelation of Several TCP Connections with Heavy Tailed Round Trip Delays

We consider that the link is shared by several TCP connections with different round trip delays, so that the round trip delays distribution is heavy tailed (see Figure 5.13). To simplify, we assume there is an infinite sequence of TCP connections and the connection with sequence number i has a round trip time equal to $D_i = Di^\beta$ for $\beta > 0$ and D fixed, while the error rate a remains the same. Of course the analysis can also be carried out with all parameters varying. In fact, a similar results about the traffic autocorrelation as the one we outline here can be obtained if we consider heavy-tailed error probabilities.

Since the TCP connections are assumed to be independent, the autocovariance

function of the aggregated traffic is equal to the sum of the autocovariance functions of the individual connections:

$$C(x) = \sum_{i=1}^{\infty} C_i(x). \quad (5.30)$$

It comes from (5.29) and the RTT distribution that:

$$C(x) \approx \frac{A}{D^2} \sum_{i=1}^{\infty} i^{-2\beta} \exp(-C \frac{x}{D} i^{-\beta}). \quad (5.31)$$

Function $C(x)$ is a harmonic sum generated from function $\exp(-C \frac{x}{D})$. The asymptotic analysis of such sums can be easily performed with the use of the Mellin transform [49]. In particular, we need to determine the definition domain and the singularity set of the Mellin transform $C^*(s)$ of function $C(x)$, defined for appropriate complex numbers s by:

$$C^*(s) = \int_0^{\infty} x^{s-1} C(x) dx. \quad (5.32)$$

From (5.31) we obtain (*cf.* [49]):

$$C^*(s) = \frac{A}{D^2} \left(\frac{C}{D} \right)^{-s} \zeta((2-s)\beta) \Gamma(s), \quad (5.33)$$

where $\zeta(s) = \sum_{i=1}^{\infty} i^{-s}$ is Euler's *zeta* function and $\Gamma(s)$ is Euler's *Gamma* function ($\Gamma(s) = \int_0^{\infty} x^{s-1} e^{-x} dx$).

The function $C^*(s)$ converges for all s such that $\Gamma(s)$ and $\zeta((2-s)\beta)$ converge. Quantity $\zeta((2-s)\beta)$ has a simple pole at $s = 2 - \frac{1}{\beta}$. Therefore $C^*(s)$ is defined on the strip $0 < \Re(s) < 2 - \frac{1}{\beta}$. The classical results on the Mellin transform state that there exist B and $\varepsilon > 0$ such that:

$$C(x) = Bx^{\frac{1}{\beta}-2} (1 + O(x^{-\varepsilon})), \quad (5.34)$$

when $x \rightarrow \infty$ [49]. This implies that the traffic autocorrelation function decays following a power law with exponent $\frac{1}{\beta} - 2$, and there are long term dependencies when $\frac{1}{2} \leq \beta \leq 1$.

However, if the number of TCP connections is finite, we expect to observe a heavy tailed behavior for a finite time scale, which is a multiple of the largest RTT in the system. In the next section, we will see that, even for a small number of connections, this upper bound can be quite large.

Simulations of Several TCP Connections

We have simulated with ns-2 the case of many TCP connections with heavy tailed RTT's sharing the same link. The simulation models 50 clients downloading very large

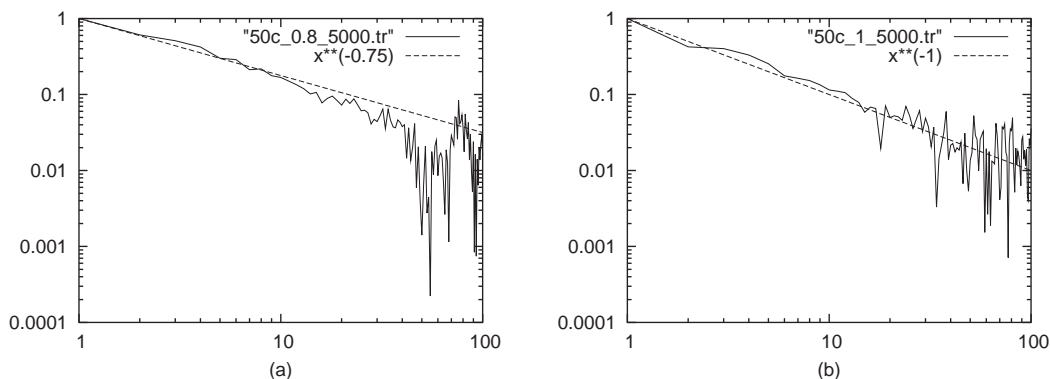


Figure 5.14: Traffic autocorrelations of 50 TCP connections with heavy tailed RTT distributions (a) $D_i = 40i^{0.8}$, (b) $D_i = 40i$. The time unit is 500ms.

files from different servers. The traffic of all connections traverses a shared link of capacity 100 Mbps. Each client is connected to the shared link via a slow link at 10 Mbps and, at the other end, each server is connected via a private link at 1 Gbps. The fixed propagation delays in these private links are chosen to follow a power law. The traffic measurements are made on the shared link in intervals of 500 ms. The packet loss rate is $a = 0.001$ for each connection. Here, the buffer losses which we described in Section 5.2.1 are emulated by a loss agent for simplicity. The packet size is 1KB and the maximum window size is 1000 packets.

In Figure 5.14 we draw the autocorrelation functions in log-log scale. The dashed line shows the theoretical power law decrease. The RTT distributions are of the form $D_i = 40i^\beta$, with $\beta = 0.8$ in (a) and $\beta = 1$ in (b). According to the theoretical analysis, the expected power law exponents for the traffic autocorrelations are $\frac{1}{\beta} - 2$, that is -0.75 and -1 , respectively.

The duration of the simulations is 5000s and the measurements start after a stabilization period of 100s.

The fluctuations in Figure 5.14 are due to the finite duration of the simulations and the fact that there are RTT's which are larger than the time unit. However, for an exact characterization of the autocorrelations in a finer time scale, one must also consider lower layer protocols as described in the previous section.

5.3 Conclusion

In this chapter, we showed that the power law distribution in service times of packets going through buffers accessing IEEE 802.11 or ethernet MAC channels generates heavy tailed traffic autocorrelation functions. Moreover, we showed via analytic means

that TCP traffic from a single connection cannot generate long term dependencies. However, several connections with heavy tailed round trip delays can generate long term dependencies. We argue that such a distribution is plausible in the Internet, as well as in large wireless networks. Finally, although the analysis was performed in the asymptotic case where the number N of parallel TCP flows tends to infinity, the simulations of the system show a good agreement with the analytical results, even when the number N is rather small. In future work, it is interesting to consider a more detailed continuous time model, which would permit to observe different properties in traffic autocorrelations in different time scales, resulting in multi-fractal characteristics in Internet traffic.

Chapter 6

Replicated Server Placement with QoS Constraints

Since communication networks and in particular the Internet have become a widely accepted medium for distributing data and all kinds of services, providers strive to satisfy user QoS requirements in a cost effective manner. A possible solution to achieve these requirements with existing protocols and networks consists in placing data servers in appropriate locations according to quality of service constraints. Therefore, the subject we study in this chapter is situated at the top of the protocol stack. We show how QoS information obtained from lower layers can be used to improve the quality of telecommunication services, as well as their organization. Within this framework, we concentrate on the problem of placing replicated data servers in various parts of the network so that the overall cost (*i.e.*, cost of opening the servers and transferring the data) is minimized, while satisfying user requirements that concern constraints on the round-trip delay of data requests. We formulate this network planning problem in a general manner, so that the algorithms we provide can be useful in many different contexts and applications, for instance in the placement of web proxies (which can also act as multicast proxies), in distributed file systems, in distributed databases, *etc.* Except from the round trip delay, we can consider other kinds of QoS requirements which can be expressed as an additive metric optimization problem, as for example the over-delay ratio, that we discussed in Chapter 2. Hence, our approach can be used in the context of wireless mesh networks, to counter the adverse effects of wireless links on delay sensitive applications.

The problem of replicated server placement has been addressed in the past in several papers. Krishnan et. al. [74] developed polynomial optimal solutions to place a given number of servers in a tree network to minimize the average retrieval cost of all clients. Li et. al. [79] investigated the placement of a limited number of Web proxies in a tree so that the overall latency for accessing the Web server is minimized. In [71] two objectives were studied: minimization of the overall access cost by all clients to access the Web site

and minimization of the longest delay for any client to access the Web site. The problem was reduced to the placement of proxies in a set of trees whose root nodes are replicas of the server. Jia et. al. [70] took the read and update operations into consideration. Qiu et. al. [93] also assumed a restricted number of replicas and no restriction on the number of requests served by each replica. A client could be served by a single replica and the cost for placing a replica was also ignored. The objective was to minimize the total cost for all clients to access the server replicas, while the cost of a request was defined as the delay, hop count or the economic cost of the path between two nodes. They compared several heuristic solutions and found that a greedy algorithm had the best performance. Chen et. al. [31],[32] tackled the replica placement problem from an other angle: minimizing the number of replicas while meeting clients' latency constraints and servers' capacity constraints by self organizing these replicas into a dissemination tree with small delay and bandwidth consumption for update dissemination. In [68] the authors considered the problem of placing a set of mirrors only at certain locations such that the maximum distance from a client to its closest mirror (server replica), based on round trip time, is minimized. They assumed no cost for placing a mirror and showed that placing mirrors beyond a certain number offered little performance gain. Sayal et. al. [96] presented a number of selection algorithms to access replicated Web servers. The algorithms found the closest replicated server for a client based on different metrics such as hop counts, round trip time and the HTTP request latency. In [102] the objective was to minimize the amount of resources, storage and update, required to achieve a certain level of service. They assumed that all servers in the network are organized into a tree structure rooted at the origin server. The construction of a distribution tree for a given set of replicas with the objective of minimizing the total communication cost of consistency management has been studied in [100]. Tang et. al. [101] presented a theoretical study on geographical replication of dynamic Web contents with the objective of minimizing the consistency management costs in terms of update transfers and object reconstruction. Cohen and Shenker [38] defined replication strategies in decentralized unstructured systems. They assumed each node had capacity ρ , which was the number of copies the node could hold and R was the total capacity of the system. Their replication strategy was a mapping from the query rate distribution to the fraction of the total system capacity allotted to each item.

In this chapter we approach the problem of replicated server placement with QoS constraints from a system administrator's perspective. Our contributions are the following. In contrast to most of the papers addressing similar problems, instead of heuristics, we provide a solution with provable performance guarantees for any possible network topology and client distribution. Also, rather than attempting to optimize metrics related to communication delays, we impose upper bounds on the delay requirements of data requests and attempt to minimize the operating cost of server placement, while respecting the delay bounds. Therefore, our approach can be adapted to different kinds of delay requirements. Moreover, in the optimization we take into account the multiplicity of server types that may be available at a site. As will be seen, the problem is NP-hard and therefore an optimal solution is not likely to be

found. We present a pseudopolynomial approximation algorithm and a polynomial time algorithm that provide guaranteed approximation factors with respect to the optimal for the problem at hand.

The rest of the chapter is organized as follows. In Section 6.1, we formulate the problem in details and we decompose it into three subproblems that can be solved independently. In Section 6.2, we present a pseudopolynomial time approximation algorithm. In Section 6.3, we provide a polynomial time algorithm with approximation factor close to the best possible (unless $\text{NP} \subseteq \text{DTIME}(n^{O(\log \log n)})$). The performance of the algorithm in simulated networks is studied in Section 6.4.

6.1 Problem Formulation

Let $G(V, E)$ represent a network with node set V , and link set E . Let also H be a subset of V . We are interested in placing servers at some of the nodes in H , that will serve requests originated by any of the nodes in V . We assume that the servers contain the same information and hence any node may obtain the requested information by accessing any of the servers.

With link (i, j) there is an associated delay d_{ij} . Requests should be obtained in a timely fashion, and hence there is a bound D on the time interval between the issuing of the request and the reception of the reply. We refer to this bound as the “round-trip delay bound”. Note that the processing time of a request at the server can be incorporated in this model by replacing D with $D - d_p$, where d_p is an upper bound on the request processing time at the server. The absolute delay values associated with each link are appropriate for wired networks, but it can be argued that they do not offer complete information for wireless links. In fact, in a wireless environment, even when the topology remains stable, the quality of the links may vary significantly. Therefore, in our problem formulation it is possible to substitute the constant link delay values with other additive metrics, as for example worst case estimates for the parameters used in Section 2.2.2 characterizing the delay distribution. In the remainder of the chapter, we will refer to absolute round trip delay bounds and we present optimized algorithms for this particular case, however it is possible to incorporate the algorithms presented in Section 2.3 in order to vary the QoS requirements.

The load in requests per second originated by node $i \in V$ is g_i . To transfer an amount of x requests per second, it is required to reserve bandwidth αx on the links traversed by the requests. The transfer of server replies corresponding to the x requests back to the requesting node, requires the reservation of βx units of bandwidth on the links traversed by the replies.

The cost of transferring 1 unit of bandwidth on link (i, j) is e_{ij} . Hence the cost of transferring x requests per second on link (i, j) is $\alpha e_{ij} x$ while the cost of transferring the replies to these requests is $\beta e_{ij} x$. A node i can split its load g_i to a number of servers

and routes as long as the delay bound D between the issuing of the request and the reception of the reply is satisfied. At each node $j \in H$ there is a set S_j of server types that can be selected. Server type s , $1 \leq s \leq K_j$, ($K_j = |S_j|$) costs f_j^s units and can process up to U_j^s requests per second.

Our objective is to determine,

1. the locations (subset of the nodes in H) where the servers will be placed,
2. the amount of traffic (in requests per unit of time) that will be routed by each node to each of the selected locations,
3. the routes through which the node traffic will be routed to each of the selected locations,
4. the type of servers and the number of each type that should be opened at each location,

so that,

1. the round-trip delay bound for each request is satisfied,
2. the total cost of using the servers and utilizing the link bandwidth is minimized.

Notice that in the current setup we do not consider link capacities. In effect we assume that the network links have enough bandwidth to carry the requested load by the network nodes. Since we consider link costs, this is a reasonable assumption, because any capacity that is required can be provided by the ISP, which in case of bandwidth saturation can either add more capacity, or charge an extra cost for the use of that link. Furthermore, in an environment where the server requests on a given link are a small portion of the total amount of information that can be supported by the network this assumption is of no consequence. However, in the case where bottleneck links could emerge we need to take special care to apply excessive costs to these links in order to avoid the concentration of too much traffic. The general problem where link capacities are also included, is a subject of further research.

6.1.1 Optimization Problem Formulation

A feasible solution to the problem consists of the following:

- A set of locations $F \subseteq H$ where the servers will be placed.
- A subset of server types $G_j \subseteq S_j$ that should be opened at location $j \in F$.

- The number n_j^s of server types $s \in G_j$ that should be opened at location $j \in F$.
- A set of round-trip routes R_{ij} between node $i \in V$ and facility $j \in F$. A round-trip route, denoted $r_{ij} = (p_{r_{ij}}, q_{r_{ij}})$, consists of two simple paths, $p_{r_{ij}}$ and $q_{r_{ij}}$. Path $p_{r_{ij}}$ originates at node i and ends at server location j , and is used for transferring requests. Path $q_{r_{ij}}$ originates at server location j and ends at node i , and is used for transferring replies.
- The amount of requests per unit of time, $x_{r_{ij}}$, accommodated on round-trip route r_{ij} .

The constraints of the problem are the following:

- The request load of each node should be satisfied. That is,

$$\sum_{j \in F} \sum_{r \in R_{ij}} x_r = g_i, \quad i \in V. \quad (6.1)$$

- The round-trip delay of each round-trip route should be at most D . That is,

$$\sum_{l \in p} d_l + \sum_{l \in q} d_l \leq D, \quad \text{for } r = (p, q) \in R, \quad (6.2)$$

where R is the set of all round-trip routes, $R = \cup_{i \in V} \cup_{j \in F} R_{ij}$, and the summation is over all links of the corresponding paths.

- The total server capacity at server location $j \in H$ should be at least as large as the request rate arriving at location j . That is,

$$\sum_{i \in V} \sum_{r \in R_{ij}} x_r \leq \sum_{s \in G_j} n_j^s U_j^s, \quad j \in H. \quad (6.3)$$

The objective cost function to be minimized is

$$\sum_{j \in F} \sum_{s \in G_j} n_j^s f_j^s + \sum_{l \in E} e_l \left(\alpha \sum_{\substack{r=(p,q) \in R \\ l \in p}} x_r + \beta \sum_{\substack{r=(p,q) \in R \\ l \in q}} x_r \right). \quad (6.4)$$

The first term in (6.4) corresponds to the cost of opening the servers, while the second term corresponds to the cost of reserving bandwidth on the network links in order to satisfy the node requests. The term involving the factor α corresponds to the bandwidth reserved on a link for transmission of node requests, while the term involving the factor β corresponds to the bandwidth reserved on the same link for transmitting replies.

From now on we assume that the node loads g_i , $i \in V$, are nonnegative integers and that splitting of these loads to a number of server locations may occur in integer units. In practice this is not a major restriction, since usually the load is measured in multiples of a basic unit.

We summarize the basic notations in Table 6.1.

Table 6.1: Notation table.

d_{ij}, e_{ij} delay and cost of link (i, j)	$F \subseteq H$ set of locations
x requests/s	$G_j \subseteq S_j$, set of server types opened at j in F
g_i load originated by node i	n_j^s number of server types $s \in G_j$ at $j \in F$
$r_{ij} = (p_{r_{ij}}, q_{r_{ij}})$ round-trip route	R_{ij} round-trip routes between i in V and j in F
D delay bound of r_{ij}	$R = \cup_{i \in V} \cup_{j \in F} R_{ij}$
$H \subseteq V$, set of possible server locations	$x_{r_{ij}}$ requests/s at r_{ij}
S_j set of possible server types at $j \in H$	f_j^s cost of server s at $j \in H$
U_j^s requests/s processed by server s at $j \in H$	$f_j(y)$ min server cost at location j for load y

6.1.2 Problem Decomposition

In this section we decompose the problem defined in Section 6.1.1 into three subproblems which can be solved independently. As will be seen all three problems are NP-hard.

For a round-trip route $r = (p, q)$, define the cost $C_r = \alpha \sum_{l \in p} e_l + \beta \sum_{l \in q} e_l$. Consider a feasible solution, π , for the optimization problem.

We can rewrite the second term in (6.4) as follows.

$$\sum_{l \in E} e_l \left(\alpha \sum_{\substack{r=(p,q) \in R \\ l \in p}} x_r + \beta \sum_{\substack{r=(p,q) \in R \\ l \in q}} x_r \right) = \sum_{i \in V} \sum_{j \in F} \sum_{r \in R_{ij}} C_r x_r. \quad (6.5)$$

Let r_{ij}^* be a minimum-cost round-trip route between node i and server location j , satisfying the round-trip delay D . Consider the feasible solution that uses the same server locations, the same servers at each location, but assigns all the request load from node i to server location j on the round-trip route r_{ij}^* , *i.e.*, it assigns on r_{ij}^* the load

$$x_{ij} = \sum_{r \in R_{ij}} x_r.$$

It follows from (6.4) and (6.5) that the new solution has cost smaller than or equal to the cost of solution π . Hence it suffices to restrict attention to solutions that assign all the load from node i to server location j , on the round-trip route r_{ij}^* . In this case, setting $c_{ij} = C_{r_{ij}^*}$, the term in (6.4) becomes $\sum_{i \in V} \sum_{j \in F} c_{ij} x_{ij}$.

Consider now the first term in (6.4). Let $f_j(y)$ be the minimum cost server type assignment at location j , under the assumption that the request load at that location is y . By definition, the feasible solution that assigns this minimum cost server assignment at location j for request load $y_j = \sum_{i \in V} \sum_{r \in R_{ij}} x_r$, is at least as good as π . Hence, we may replace this term with $\sum_{j \in F} f_j(y_j)$.

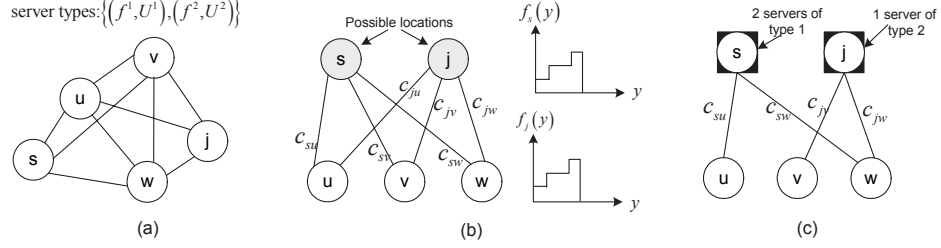


Figure 6.1: a) A graph with five clients, $V = \{s, u, v, w, j\}$, two possible locations $H = \{s, j\}$ and two types of servers for each location. b) The modified graph where each link represents the round-trip minimum cost routes, satisfying the delay constraint between a client and a possible location. Cost functions $f_s(s)$ and $f_j(y)$ of the possible locations. c) The resulting graph. Servers have been placed at the appropriate locations.

For our purposes, it is important to observe that the function $f_j(y)$ defined in the previous paragraph is subadditive, *i.e.*, it satisfies the inequality

$$f_j(y_1) + f_j(y_2) \geq f_j(y_1 + y_2) \text{ for all } y_1 \geq 0, y_2 \geq 0. \quad (6.6)$$

To see this, note that if $S(y_1), S(y_2)$ is the set of servers achieving the optimal costs $f_j(y_1), f_j(y_2)$ respectively, then the set of servers $S(y_1) \cup S(y_2)$ provides a feasible solution for request load $y_1 + y_2$, with cost $f_j(y_1) + f_j(y_2)$. Since $f_j(y_1 + y_2)$ is by definition the minimum cost server assignment with request load $y_1 + y_2$, (6.6) follows.

From the discussion above it follows that we need to solve the following problems.

Problem 1. Given a graph, find a round-trip route with minimum cost, satisfying the round-trip delay bound for any node $i \in V$ and server location $j \in H$. This determines c_{ij} , $i \in V, j \in H$.

Problem 2. Given a set of server types S_j and a required load y at node $j \in H$, find the optimal selection of server types and the number of servers $n_j^s(y)$ of each type s so that the load y is accommodated. That is, determine $n_j^s(y)$ so that $\sum_{s \in G_j} n_j^s(y) U_j^s \geq y$ and $f_j(y) = \sum_{s \in G_j} n_j^s(y) f_j^s$ is minimal.

Problem 3. Given non-decreasing subadditive functions $f_j(y)$, costs c_{ij} , integer node loads $g_i \geq 0, i \in V$, solve

$$\begin{aligned} & \min \sum_{j \in H} f_j(y) + \sum_{i \in V} \sum_{j \in H} c_{ij} x_{ij} \\ & \text{subject to: } \sum_{j \in H} x_{ij} = g_i, i \in V, \sum_{i \in V} x_{ij} = y, j \in H, x_{ij} \geq 0. \end{aligned}$$

A graphical representation of the problem is depicted in Figure 6.1.

The decision problems associated to Problems 1 and 2 are NP-hard. Indeed, when $\beta = 0$, the associated decision problem to Problem 1 is reduced to the Shortest Weight-Constrained Path problem which is known to be NP-hard [55]. Also, the associated decision problem to Problem 2 amounts to the Unbounded Knapsack Problem which is NP-hard [72]. However for both problems pseudopolynomial algorithms exist (see Section 6.2) and, as will be discussed in Section 6.3, fully polynomial time approximation algorithms can be developed. Regarding Problem 3, there is an extensive work in the literature under various assumptions on the function $f_j(y)$ and on the costs c_{ij} ([35], [73], [23], [67], [89], [34], [59]). Most of the work is concentrated on the case of “metric” costs, *i.e.*, it is assumed that costs satisfy the inequality $c_{ij} + c_{jk} \geq c_{ik}$. However, this inequality does not hold in our case. Moreover, $f_i(y)$ is assumed computable at unit cost while in our case $f_i(y)$ cannot be computed in polynomial time (unless $P = NP$).

In the next section, by combining algorithms for the three problems discussed above, we provide a pseudopolynomial time approximation algorithm for the problem addressed in this thesis. The algorithm for Problem 3 is based on the algorithm proposed in [59] and uses the fact that $f_j(y)$ is a subadditive step function. In Section 6.3 we modify the algorithm in order to obtain a polynomial time algorithm with approximation factor close to the best possible (unless $NP \subseteq DTIME(n^{O(\log \log n)})$).

6.2 Pseudopolynomial Algorithm

In this section we discuss pseudopolynomial algorithms for each of the Problems 1, 2 and 3. By combining these algorithms we get a pseudopolynomial algorithm for the problem at hand.

6.2.1 Pseudopolynomial Algorithm for Problem 1

This problem is a multi-metric optimization problem as the one described in Section 2.3.2. Therefore, it can also be solved using dynamic programming. In fact, in case we consider over-delay ratio requirements, the second algorithm presented in Section 2.3.2 can be used in this context. However, the adaptation to general link costs (instead of the hop count for the first metric) makes it also pseudopolynomial.

We will now discuss the fixed delay case in more detail.

Let $F_{ij}(d)$ be the value of the minimum cost (forward) path from node i to j with delay at most d , and $B_{ij}(d)$ the value of the minimum cost (backward) path from node j to i with delay at most d . Here, for the computation of forward and backward paths the link costs are taken as αe_{ij} , βe_{ij} respectively. Then it can be easily seen that,

$$c_{ij} = \min_{0 \leq d \leq D} \{F_{ij}(d) + B_{ij}(D - d)\} \quad (6.7)$$

Based on (6.7), c_{ij} can be determined provided $F_{ij}(d)$ and $B_{ij}(d)$ are known. There

are fully polynomial time, generally complex, algorithms for computing these quantities. In this section we will concentrate on efficient pseudopolynomial algorithms that work well in practice [97],[85]. We provide the discussion for $F_{ij}(d)$, since the same holds for $B_{ij}(d)$. The algorithms in [97],[85] are based on the fact that $F_{ij}(d)$ is a right continuous non-increasing step function with a finite number of jumps. Hence, in order to compute $F_{ij}(d)$ one needs only to compute its jumps, which in several practical networks are not many. Another useful feature of these algorithms is that in one run they compute $F_{ij}(d)$ from a given node $j \in H$ to all other nodes in V .

Let K_{ij}^f be the number of jump points of $F_{ij}(d)$ and K_{ij}^b be the number of jump points of $B_{ij}(d)$. Let $d_{ij}^f(k)$, $1 \leq k \leq K_{ij}^f$, $i \in V$ be the jump points of $F_{ij}(d)$ such that $d_{ij}^f(k-1) < d_{ij}^f(k) \leq D$, $k = 2, \dots, K_{ij}^f$. Similarly, let $d_{ij}^b(k)$, $1 \leq k \leq K_{ij}^b$, $i \in V$ be the jump points of $B_{ij}(d)$. The optimal round-trip costs c_{ij} , $j \in H$, $i \in V$ can be computed using Algorithm 3. The jumps in steps 2 and 3 can be computed using the algorithm in [97]. The “for” loop in step 6 implements the minimization required by (6.7), taking into account that $F_{ij}(d)$ and $B_{ij}(d)$ are step functions.

Algorithm 3 Algorithm for finding the minimum cost round-trip route

Input: Graph G with link costs and delays, round-trip delay bound D .

Output: The array c with the costs of the round-trip routes.

1. For any node j in H do
 2. Compute jump points of $F_{ij}(d)$, $d_{ij}^f(k)$, $1 \leq k \leq K_{ij}^f$, $i \in V$
 3. Compute jump points of $B_{ij}(d)$, $d_{ij}^b(k)$, $1 \leq k \leq K_{ij}^b$, $i \in V$
 4. For $i \in V$ do
 5. $c_{ij} = \infty$
 6. For $k = 1$ to K_{ij}^f do
 7. Let d_{ij}^b be the largest jump point of $B_{ij}(d)$ not exceeding $D - d_{ij}^f(k)$
 8. $c_{ij} \leftarrow \min \left\{ c_{ij}, F_{ij}(d_{ij}^f(k)) + B_{ij}(d_{ij}^b) \right\}$
-

We now discuss the complexity of Algorithm 3. For the purposes of complexity analysis, in the rest of the chapter we will assume that in the worst case $H = V$. Using the algorithm and the analysis presented in [97] it can be proved that the worst case running time for the computation of the jump points for all nodes in H is $O(|V|D(|V| \log |V| + |E| \log |V|))$. The running time of the minimum operation (line 8)

is $O(|V|^2 D)$. Thus the running time of this algorithm is dominated by the time needed to compute the jump points.

6.2.2 Pseudopolynomial Algorithm for Problem 2

We restate Problem 2 in its generic form, to simplify notation.

Problem 2 (generic form). Given a set of server types S , server capacities U^s , server costs $f^s > 0$ and a required load y , find the optimal selection of server types G and the number of servers of each type so that the load is satisfied. That is, determine $n^s(y)$ so that $\sum_{s \in G} n^s(y) U^s \geq y$ and $f(y) = \sum_{s \in G} n^s(y) f^s$ is minimal.

Problem 2 is similar to the Unbounded Knapsack Problem (UKP) [72]. The difference is that in UKP the inequality constraint is reversed and maximization of the cost $\sum_{s \in G} n^s(y) f^s$ is sought. A pseudopolynomial algorithm for Problem 2 can be developed in a manner analogous to the one used for UKP, using dynamic programming. Specifically, number the servers from 1 to $|S|$ and define $A(f, i)$ to be the largest load achievable by using some among the first i servers so that their total cost is f . The entries of the table $A(f, i)$ can be computed in order of increasing i and f using the dynamic programming equation

$$A(f, i + 1) = \min\{A(f, i), U^{i+1} + A(f - f^{i+1}, i + 1)\}, \quad (6.8)$$

with $A(f, 0) = 0$ for all f , $A(f, i) = -\infty$ if $f < 0$, and $A(0, i) = 0$ for all $0 \leq i \leq K$. The optimal server selection cost is then determined as $f(y) = \min\{f \mid A(f, K) \geq y\}$. By keeping appropriate structures one can also determine the server types and the number of servers of each type for achieving the optimal solution.

The function $f(y)$ has properties similar to those of $F_{ij}(d)$ and $B_{ij}(d)$. Specifically, it is a right continuous non-decreasing step function. Moreover, based on (6.8) and using an approach similar to [21], an efficient pseudopolynomial algorithm can be developed for finding the jump points of $f(y)$. Again, an important property in our case is that in one run of the algorithm, all jump points of $f(x)$, for integer $x \leq y$, can be determined. The running time of this approach can be bounded by $O(|S| y)$, where $|S|$ is the number of server types.

6.2.3 Pseudopolynomial Algorithm for Problem 3

In [59] a polynomial time algorithm is provided for Problem 3 for the case of concave facility cost functions. It is assumed that the cost $f_j(y)$ of placing servers at node $j \in H$ to accommodate load y can be computed at unit cost and that all nodes have unit loads. It is shown that the proposed algorithm achieves an approximation factor of $\ln |V|$ compared to the optimal. In our case we have arbitrary integer node loads g_i while the functions $f_j(y)$ are subadditive and can be computed exactly only in pseudopolynomial

time. As observed in [82] the assumption of unit loads can be removed by considering a modified network where node i is replaced with g_i nodes each having the same costs to nodes in H as node i has. However, now the algorithm becomes pseudopolynomial (even assuming unit costs for computing $f_j(y)$) since the number of nodes in the modified network can be as large as $|V|g_{\max}$, where $g_{\max} = \max_{i \in V} \{g_i\}$.

The approximability proof for general costs c_{ij} in [59] carries over without modification if $f_j(y)$ are subadditive rather than concave functions. Hence the approximability factor in our case becomes $\ln(|V|g_{\max})$.

To our knowledge, the algorithm in [59] is the only one proposed in the literature, that can provide performance guarantees in terms of approximability to the optimal for general costs c_{ij} . Moreover, its worst case running time is among the best of the proposed algorithms. Hence, we will use the algorithm in [59] as the basis for our development. We present it below (Algorithm 4) adapted to our situation. For the moment we assume that $f_j(y)$ can be computed exactly.

The algorithm performs a number of iterations. At each iteration a node j^* in H is selected and the load of some of the nodes in V is assigned to j^* . Let matrix $\psi(i, j)$ represent the total load from node i assigned to server location j at the beginning of an iteration (*i.e.*, the beginning of the while loop at step 3). Hence the load of node i remaining to be assigned is $r(i) = g_i - \sum_{j \in H} \psi(i, j)$.

A node such that $r(i) > 0$ is called unassigned. For server location $j \in H$ consider the unassigned nodes arranged in non-decreasing order of their costs $c_{i_s j}$, *i.e.*, $c_{i_1 j} \leq c_{i_2 j} \leq \dots \leq c_{i_m j}$. Let $R_j(n) = \sum_{s=1}^n r(i_s)$, $1 \leq n \leq m$, and $n_j(k) = \max\{n : R_j(n) \leq k\}$. Define also $l_j(k) = k - R_j(n_j(k))$.

The variable $load_j$ holds the total load assigned to node $j \in H$ at the beginning of an iteration. In step 5, the most economical (cost per unit of assigned load) load assignment for each of the server locations is computed. In steps 7 and 8, the server location with the minimum economical assignment is selected and the associated load is placed on this location. In steps 9 to 13, updating of the remaining loads of the nodes that will place their load on the selected server location is taking place.

The average running time of this algorithm can be improved by taking advantage of the fact that in this case $f_j(y)$ is a step function. Specifically, in the this chapter's appendix it is shown that in order to compute the minimum in step 5, one needs to do the computation only for values of k such that $load_j + k$ is a jump point of $f_j(y)$, or $k = R_j(n)$ for some n .

In the appendix, we also show that with the use of appropriate data structures and assuming unit cost for computing $f_j(y)$, the running time of Algorithm 4 is

$$O(|V|^3 g_{\max}^2). \quad (6.9)$$

Letting $|S|$ be the maximum number of server types in any of the server locations,

Algorithm 4 Generic algorithm for solving Problem 3

Input: Graph G , the array c with the costs of the routes and the Knapsack list.

Output: Locations and types of servers, routes and load assigned from each client to the selected locations.

1. For $j \in H$ set $load_j = 0$
2. For $i \in V$ set $\psi(i, j) = 0$
3. While there is an unassigned node do
4. For $j \in H$ do
5.
$$t(j) = \min_k \frac{f_j(load_j+k) - f_j(load_j) + \sum_{s=1}^{n_j(k)} r(i_s)c_{i_s j} + l_j(k)c_{i_{n_j(k)+1}j}}{k}$$
6.
$$k(j) = \arg \min_k \frac{f_j(load_j+k) - f_j(load_j) + \sum_{s=1}^{n_j(k)} r(i_s)c_{i_s j} + l_j(k)c_{i_{n_j(k)+1}j}}{k}$$
7. Let $j^* = \arg \min_{j \in H} \{t(j)\}$, $k^* = k(j^*)$
8. Set $load_{j^*} \leftarrow load_{j^*} + k^*$
9. For $1 \leq s \leq n_{j^*}(k^*)$ do
10. $\psi(i_s, j^*) \leftarrow \psi(i_s, j^*) + r(i_s)$
11. $r(i_s) = 0$
12. $\psi(i_{n_{j^*}(k^*)+1}, j^*) = l_{j^*}(k^*)$
13. $r(i_{n_{j^*}(k^*)+1}) \leftarrow r(i+1) - l_{j^*}(k^*)$

taking into account that the maximum load on any facility is $|V|g_{\max}$ and that we may need to compute at most $|V|$ functions $f_j(y)$, we conclude that the worst case computation time of the complete algorithm is

$$O\left(|V|D(|V|\log|V| + |E|\log|V|) + |S||V|^2g_{\max} + |V|^3g_{\max}^2\right). \quad (6.10)$$

The term $|V|D(|V|\log|V| + |E|\log|V|)$ corresponds to the computation of c_{ij} . The term $|S||V|^2g_{\max}$ corresponds to the cost of computing $f_j(|V|g_{\max})$, $j \in H$. Note that as mentioned in Section 6.2.2, for each $j \in H$, computing $f_j(|V|g_{\max})$ also computes all the jump points of $f_j(y)$, $y \leq |V|g_{\max}$. As a result, when implementing Algorithm 4, $f_j(y)$ can be computed at unit cost. Hence the third term in (6.10) represents the worst case computation time for running Algorithm 4. As mentioned in Section 6.2.3 the

approximability factor of this algorithm is $\ln(|V|g_{\max})$.

As will be seen in Section 6.4 the proposed algorithm works well in practice. However, since (6.10) involves the input parameters D and g_{\max} , the proposed algorithm is pseudopolynomial. It is theoretically important to know whether there exists a polynomial time algorithm that can provide a guaranteed approximation factor with respect to the optimal. In the next section we will show that this can be done based on the algorithm presented above.

6.3 Polynomial Algorithm

In this section, by generalizing the approach in [59] we provide a polynomial time approximation algorithm for arbitrary integer node loads and non-decreasing subadditive functions that are not necessarily computable exactly in polynomial time. Note that a concave function is also subadditive and hence our results carry over to concave functions. However, as will be seen, for concave functions the approximation constants can be made smaller.

The approach we follow is to provide polynomial time approximation algorithms for each of Problems 1, 2 and 3. By combining these algorithms, we get a polynomial time algorithm for the problem at hand with guaranteed performance factor compared to the optimal.

In the previous section the costs c_{ij} and the functions $f_j(y)$ were computed exactly using pseudopolynomial algorithms for Problems 1 and 2 respectively. The use of polynomial time approximation algorithms for these problems provides only approximate values for c_{ij} and $f_j(y)$. That is, we can only ensure that for any $\varepsilon > 0$, we provide in polynomial time values \bar{c}_{ij} and $\bar{f}_j(y)$ (for fixed y) such that $c_{ij} \leq \bar{c}_{ij} \leq (1 + \varepsilon)c_{ij}$ and $f_j(y) \leq \bar{f}_j(y) \leq (1 + \varepsilon)f_j(y)$, $y \geq 0$. Replacing c_{ij} and $f_j(y)$ with \bar{c}_{ij} and $\bar{f}_j(y)$ in Problem 3 and providing an a -approximate solution for the resulting instance, provides also an $(1 + \varepsilon)a$ -approximate solution for the original problem. This important observation was used in [82] and we present it here in the next lemma.

Lemma 1 *Consider the problems*

$$\min_{\mathbf{x} \in A} g(\mathbf{x}), \quad A \subseteq R^n, \quad (6.11)$$

$$\min_{\mathbf{x} \in A} \bar{g}(\mathbf{x}), \quad A \subseteq R^n, \quad (6.12)$$

where $g(\mathbf{x}) \geq 0$. If for $\mathbf{x} \in A$,

$$g(\mathbf{x}) \leq \bar{g}(\mathbf{x}) \leq bg(\mathbf{x}), \quad (6.13)$$

then an a -approximate solution for problem (6.12), $a \geq 1$, is a ba -approximate solution for (6.11).

Proof: See Appendix.

Using Lemma 1 we can proceed as follows.

- Compute in polynomial time approximate values \bar{c}_{ij} ,
- Compute in polynomial time approximate values $\bar{f}(y)$ (for a given $y \geq 0$),
- Provide an approximation algorithm for Problem 3, based on Algorithm 4, using the approximate values \bar{c}_{ij} , $\bar{f}(y)$.

Difficulties arise in the approach outlined above for the following reasons. First, to compute the minimum in step 5 of Algorithm 4, $\bar{f}(y)$ must be computed for all values of y in the worst case, and the number of these computations is bounded by $|V|g_{\max}$, *i.e.*, it is not polynomial in the input size, even if $\bar{f}(y)$ is computable at unit cost. Second, the amount of load assigned to a node in H at each iteration of the “for” loop at step 9 can be 1 in the worst case and hence the number of iterations of the while loop may be again $|V|g_{\max}$ in the worst case. Third, while $f(y)$ is subadditive, it cannot be guaranteed that $\bar{f}(y)$ is subadditive as well, and hence the approximation factor with respect to the optimal cannot be guaranteed a priori. Hence, the straightforward application of Algorithm 4 will result in pseudopolynomial worst case running time and will not provide us with guaranteed performance bounds. As will be seen, however, we can modify the approach so that the resulting algorithm runs in polynomial time at the cost of a small increase in the approximation factor.

6.3.1 Polynomial Algorithms for Problems 1 and 2

A fully polynomial time approximation algorithm for the problem of finding the minimum constrained path from a source to a given destination was developed by Hassin [60]. An improvement of this algorithm was presented in [81]. The approach in [81] consists in defining a test procedure, which is used iteratively to find upper and lower bounds for the restricted shortest path. The latter algorithm can be modified in order to develop a fully polynomial time approximation algorithm for Problem 1, that is finding a constrained round trip path. The adaptation consists in considering bounds for round trip paths and modifying Algorithm SPPP in [81], which constitutes the test procedure mentioned before. The details on the adaptation can be found in the technical report [10]. The resulting algorithm runs in $t = O(|E||V|(\log \log |V| + 1/\varepsilon))$ for each pair of (client, server) nodes.

A fully polynomial time approximation algorithm for Problem 2 can be developed by paralleling the approach for the UKP [72, Section 8.5]. The resulting algorithm has a worst case running time of $T = O(\frac{1}{\varepsilon^2} |S| \log |S|)$, where $|S|$ is the number of server types.

6.3.2 Polynomial Algorithm for Problem 3

We now address the main problem of Section 6.3, *i.e.*, the development of a polynomial algorithm for Problem 3, using the approximate costs \bar{c}_{ij} and $\bar{f}_j(y)$. We intend to use Algorithm 4 as the basis for the development. Assume for the moment that c_{ij} and $f_j(y)$ are computable exactly. As was mentioned above the fact that the node loads are general nonnegative integers in our case, renders the algorithm pseudopolynomial even under this assumption. However, if the functions $f_j(y)$ are concave, then the algorithm becomes polynomial. This is due to the fact that for concave functions Algorithm 4 can assign all the load of each node to a single server. This is shown in the next lemma. Recall that a function $f(y)$ defined for integer y is called concave if for all y in its domain of definition it holds, $f(y+1) - f(y) \leq f(y) - f(y-1)$.

Lemma 2 *If the functions $f_i(y)$ are concave and Algorithm 4 is applied, then the load of each node in V can be assigned to a single server.*

Proof: See Appendix.

Lemma 2 implies that when c_{ij} and $f_j(y)$ are computable in polynomial time and $f_j(y)$ are concave, the algorithm runs in polynomial time. Indeed, if t is the worst case computation time needed to compute ε -approximate \bar{c}_{ij} , $i, j \in V$, according to the algorithm used to solve Problem 1, then the time needed to compute the $|V|^2$ values of \bar{c}_{ij} is $O(|V|^2 t)$. Next, using the \bar{c}_{ij} as input to Algorithm 4, letting T be the worst case computation time needed to compute ε -approximate $\bar{f}_j(y)$ according to the algorithm used to solve Problem 2 and following a similar reasoning as in the proof of (6.9), it can be seen that the worst case running time of the Algorithm 4 is $O(|V|^3 T)$. Hence the computation time for the whole algorithm is $O(|V|^3 T + |V|^2 t)$ and the approximation factor is $(1 + \varepsilon) \log(|V| g_{\max})$. Since as discussed in Section 6.3.1 both t and T are polynomial for given ε , the resulting algorithm is also polynomial.

We now return to the problem at hand. In our case, c_{ij} and $f_j(y)$ are not computable in polynomial time and, moreover, $f_i(y)$ is subadditive rather than concave. Hence the results above cannot be applied directly to obtain a polynomial time algorithm. Regarding c_{ij} , as discussed in Section 6.3 we replace c_{ij} with polynomially computable approximations \bar{c}_{ij} . This takes time $O(V^2 t)$. Dealing with $f_j(y)$ requires more care since, as it is also discussed in 6.3, simply approximating $f_j(y)$ for each y results in pseudopolynomial time algorithm and in no approximation factor guarantees. The approach we follow is to construct in polynomial time a concave function $\tilde{f}_j(y)$, such that for any y in its domain of definition $\tilde{f}_j(y)$ is computed in polynomial time and,

$$f_j(y) \leq \tilde{f}_j(y) \leq a f_j(y). \quad (6.14)$$

Then, by applying Lemmas 1 and 2 we get a polynomial time algorithm. To proceed we need some definitions.

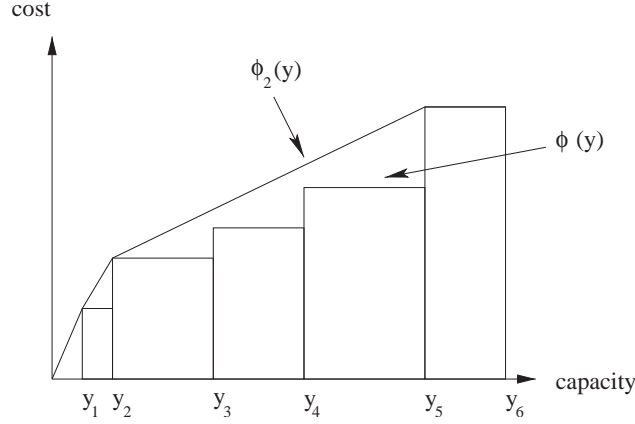


Figure 6.2: A subadditive step function $\phi(y)$ with its jump points y_1, \dots, y_6 and its upper hull $\phi_2(y)$.

Consider a nonnegative function $\phi : \{0, 1, \dots, W\} \rightarrow \mathbb{Q}^+$ (\mathbb{Q}^+ is the set of nonnegative rationals) and let A be the convex hull of the set of points $S = \{(y, \phi(y)), y = 0, 1, \dots, W\} \cup \{(0, 0), (W, 0)\}$. Recall that the convex hull of a set of points S is the smallest convex set that includes these points. In two dimensions it is a convex polygon. The vertices of the polygon correspond to a subset of S , of the form $S' = \{(y_k, \phi(y_k)), k = 1, \dots, K\} \cup \{(0, 0), (W, 0)\}$ where $y_k \in \{0, 1, \dots, W\}$, $y_1 = 0$, $y_K = W$, and $y_k < y_{k+1}$ for all k , $1 \leq k \leq K - 1$.

Consider the piecewise linear function $\phi_2(y)$ with break points the set S' , *i.e.*, for $y_k \leq y < y_{k+1}$, $\phi_2(y)$ is defined as,

$$\phi_2(y) = \phi(y_k) + \frac{\phi(y_{k+1}) - \phi(y_k)}{y_{k+1} - y_k}(y - y_k). \quad (6.15)$$

The function $\phi_2(y)$ is concave. We call $\phi_2(y)$, the “upper hull” of $\phi(y)$. An example of a subadditive step function and its upper hull is depicted in Figure 6.2. If $\phi(y)$ is non-decreasing, then $\phi_2(y)$ is also non-decreasing. By construction it holds for all $y \in \{0, 1, \dots, W\}$,

$$\phi(y) \leq \phi_2(y). \quad (6.16)$$

As the next lemma shows, if the function $\phi(y)$ is subadditive and non-decreasing, then it also holds that its upper hull is at most $2\phi(y)$.

Lemma 3 *If a function $\phi : \{0, 1, \dots, W\} \rightarrow \mathbb{Q}$ is subadditive and non-decreasing, then it holds for its upper hull $\phi_2(y)$, $\phi_2(y) \leq 2\phi(y)$.*

Proof: See Appendix.

Consider now the subadditive function $f(y)$ of interest in our case (we drop the index j for simplicity). As a consequence of the approximate solution to Problem 3, for a given $\varepsilon > 0$ and a given $y \in \{0, 1, \dots, W\}$, $W = |V| g_{\max}$, we can construct in polynomial time a non-decreasing function $\bar{f}(y)$ such that

$$f(y) \leq \bar{f}(y) \leq (1 + \varepsilon)f(y). \quad (6.17)$$

Let $\bar{f}_2(y)$ be the upper hull of $\bar{f}(y)$. By (6.17), $\bar{f}_2(y)$ is smaller than or equal to the upper hull of $(1 + \varepsilon)f(y)$, which in turn by Lemma 3 is smaller than $2(1 + \varepsilon)f(y)$ (notice that $(1 + \varepsilon)f(y)$ is subadditive). Hence we will have

$$f(y) \leq \bar{f}_2(y) \leq 2(1 + \varepsilon)f(y). \quad (6.18)$$

Since $\bar{f}_2(y)$ is concave, if we replace c_{ij} with \bar{c}_{ij} and $f(y)$ with $\bar{f}_2(y)$, we can provide an approximate solution to Problem 2 with approximation factor $\log(|V| g_{\max})$. From (6.18) and Lemma 1 we will then have a solution to our original problem with approximation factor $2(1 + \varepsilon) \log |V g_{\max}|$.

The problem that remains to be solved is the construction of the upper hull of $\bar{f}(y)$ in polynomial time. There are at most $W' = W + 2$ points in the set $\{(y, \bar{f}(y)), y = 0, 1, \dots, W\} \cup \{(0, 0), (W, 0)\}$ and the upper hull of the points in this set can be constructed (*i.e.*, its break points can be determined) in time $W' \log W'$ [42]. However, in our case $W = |V| g_{\max}$ and hence the straightforward construction of the upper hull requires pseudopolynomial construction time. To address the latter problem, we construct first a non-decreasing step function $\hat{f}_1(y)$ with polynomial number of jump points (y is a jump point of $\hat{f}_1(y)$ if $\hat{f}_1(y - 1) \neq \hat{f}_1(y)$) that is a good approximation to $\bar{f}(y)$, and then we construct the upper hull of $\hat{f}_1(y)$. Since $\hat{f}_1(y)$ has polynomial number of jump points its upper hull will also have polynomial number of break points and can be constructed in polynomial time.

Algorithm 5

Input: Algorithm for computing $\bar{f}(y)$, $\varepsilon > 0$.

Output: The sequence, \hat{f}_k , $k = 0, 1, \dots, K$.

1. $\hat{f}_0 = 0$, $\hat{f}_1 = \bar{f}(1)$, $y_1 = 1$, $k = 2$,
 2. $\hat{f}_k = (1 + \varepsilon)\bar{f}(y_{k-1})$
 3. If $\hat{f}_k > \bar{f}(W)$, set $y_k = W$, $K = k$ and stop. Else,
 4. Determine y_k such that $\bar{f}(y_k - 1) \leq \hat{f}_k \leq \bar{f}(y_k)$
 5. $k = k + 1$, go to step 2.
-

We have by definition $\bar{f}(0) = 0$, $\bar{f}(1) = f(1) > 0$. Consider the sequence of integers $\hat{f}_0 = 0$, \hat{f}_k , $k = 1, \dots, K$, generated by Algorithm 5. The sequence \hat{f}_k , $k = 0, 1, \dots, K$ can be used to construct a step function that is a good approximation to $\bar{f}(y)$. This is shown in the next lemma.

Lemma 4 a) In Algorithm 5, $K = O\left(\frac{1}{\epsilon} \log \frac{\bar{f}(W)}{\bar{f}_1}\right)$.

b) The worst case running time of Algorithm 5 is $O\left(T \log(W) \frac{1}{\epsilon} \log \frac{\bar{f}(W)}{\bar{f}_1}\right)$, where T is the worst case time (over all y , $1 \leq y \leq W$) needed to compute $\bar{f}(y)$.

c) Consider the step function defined as follows: if $y_k \leq y < y_{k+1}$ for some k , $1 \leq k \leq K - 1$, then $\hat{f}(y) = \hat{f}_k$, and $\hat{f}(W) = \hat{f}_K$. It holds,

$$\hat{f}(y) \leq \bar{f}(y) \leq (1 + \epsilon)\hat{f}(y) \quad (6.19)$$

Proof: See Appendix.

Based on (6.19), we can use $\tilde{f}(y) = (1 + \epsilon)\hat{f}(y)$ as a function to approximate $f(y)$. This is shown in the next lemma.

Lemma 5 Let $\epsilon_0 > 0$, $\epsilon > 0$ be given. Let $f(y)$ be the optimal solution to Problem 2 and assume that we compute for a given y the approximate function $\bar{f}(y)$ so that

$$f(y) \leq \bar{f}(y) \leq (1 + \epsilon_0) f(y). \quad (6.20)$$

a) For the purposes of computing the step function $\hat{f}(y)$ satisfying (6.19), $\bar{f}(y)$ may be assumed non-decreasing.

b) It holds for $\tilde{f}(y) = (1 + \epsilon)\hat{f}(y)$

$$f(y) \leq \tilde{f}(y) \leq (1 + (\epsilon + \epsilon_0 + \epsilon\epsilon_0)) f(y), \quad (6.21)$$

c) The number of jump points of $\hat{f}(y)$, hence of $\tilde{f}(y)$, is $O\left(\frac{1}{\epsilon} \log(|V| g_{\max})\right)$ and the running time of Algorithm 5 is $O\left(\frac{1}{\epsilon} T (\log(|V| g_{\max}))^2\right)$, where T is the worst case time (over all y , $1 \leq y \leq W$) needed to compute $\bar{f}(y)$.

Proof: See Appendix.

From the discussion above we have polynomial time Algorithm 6 for computing the server locations.

In the algorithm, for simplicity, we pick a single ϵ for all the approximations. If needed, a separate ϵ can be used for each of the approximations.

Let $|S| = \max_{j \in H} \{|S_j|\}$. Recall that $\phi_j(y)$, $j \in H$ are non-decreasing piecewise linear functions with at most K number of break points, where $K = O(\log(|V| g_{\max})/\epsilon)$.

Algorithm 6 Polynomial Time Algorithm For Calculating Server Locations

Input: Polynomial Algorithm for Problem 1, Algorithms 4 and 5, $\epsilon > 0$.

Output: Locations and types of servers, routes and load assigned from each client to the selected locations.

1. For $i \in V, j \in H$, compute \bar{c}_{ij} so that $c_{ij} \leq \bar{c}_{ij} \leq (1 + \epsilon)c_{ij}$, $i \in V, j \in H$.
 2. For $j \in H$, construct the step functions $\hat{f}_j(y)$, from $\bar{f}_j(y)$ according to Algorithm 5, using as subroutine the algorithm for computing, for a given $y > 0$, $\bar{f}_j(y)$ such that $f_j(y) \leq \bar{f}_j(y) \leq (1 + \epsilon)f_j(y)$.
 3. Construct the upper hull of $\tilde{f}_j(y) = (1 + \epsilon)\hat{f}_j(y)$. Let $\phi_j(y)$ be this upper hull.
 4. Use Algorithm 4 to solve Problem 3, where c_{ij} is replaced by \bar{c}_{ij} and $f_j(y)$ is replaced by $\phi_j(y)$.
-

In fact, the computation of $\phi_j(y)$ in step 3 of Algorithm 6 amounts to storing the break points of $\phi_j(y)$. Hence it takes time $O(\log K)$, in the worst case, to compute $\phi_j(y)$ for a given y and according to the discussion in Section 6.3.2 it takes time $O(|V|^3 \log K)$ to execute Algorithm 4 using $\phi_j(y)$.

Taking into account the previous discussion, the worst case running times of each step are:

1. $O(|V|^2 t) = O(|E||V|^3(\log \log |V| + 1/\epsilon))$
2. $O(|V| \frac{1}{\epsilon} T(\log(|V| g_{\max}))^2) = O(\frac{1}{\epsilon^3} |V| |S| \log |S| (\log(|V| g_{\max}))^2)$
3. $O(K \log K) = O(\frac{1}{\epsilon} \log(|V| g_{\max}) \log(\log(|V| g_{\max})/\epsilon))$
4. $O(|V|^3 \log K) = O(|V|^3 \log(\log(|V| g_{\max})/\epsilon))$

The resulting algorithm has a guaranteed performance ratio of $2(1 + \epsilon)^2 \log(|V| g_{\max})$ and its worst case running time is dominated by steps 1 and 2:

$$O(|E||V|^3(\log \log |V| + 1/\epsilon) + \frac{1}{\epsilon^3} |V| |S| \log |S| (\log(|V| g_{\max}))^2).$$

Note that since in Algorithm 6 the functions $\phi_j(y)$ are concave by construction, by Lemma 2 the algorithm assigns all the load of each node $i \in V$ to a single server node in H .

6.4 Numerical Results

In this section we evaluate the proposed algorithm using sample topologies following the power law model, *i.e.*, where the node degrees follow a power law distribution (such a distribution has been observed in the Internet). These topologies are taken from the BRITE package [83] by using the Router Barabasi-Albert Model. We used random parameters for the network, rather than specific application parameters in order to test the overall performance of the algorithm under general conditions. For each network the delay d_l of a link is picked randomly with a uniform distribution among the integers $[1, 100]$ and the cost is generated in such a manner that it is correlated to its delay. Thus, for each link l a parameter b_l is generated randomly among the integers $[1, 5]$. The cost of link l is then $b_l(101 - d_l)$. Hence the link cost is a decreasing function of its delay. We assume that the the same server types can be placed in each of the locations. For our simulations we use 4 different server types with capacities and costs equal to $\{(100, 3000) (150, 3500) (250, 4000) (350, 5000)\}$ respectively. We set the factors $\alpha = 0.1$ and $\beta = 0.2$ and assume the load in requests per unit of time originated by each node is randomly chosen among the integers $[1, 100]$. We also assume $H = V$, *i.e.*, servers may be placed in any of the nodes.

We are not aware of other approaches in the literature addressing the form of the problem examined in this thesis. Hence direct comparison of our proposal to other algorithms cannot be made. However, since there are several proposals that concentrate on metrics related to optimizing the delays of requests in some manner, we examine the performance of the latter algorithms in case operating costs were involved. Specifically, we run two sets of experiments which differ in the manner the round-trip routes for requests are selected.

In the first experiment ten different power law network topologies are generated with $|V| = 100$ nodes and $|E| = 970$ edges. We run the algorithm for 6 different delay constraints $D = \{100, 200, 400, 500, 600, 800\}$.

MinDelay: This manner of selecting routes has been employed in [93] and [96] where all requests from client node i to server node j are send over the minimal delay round-trip route. Hence in this algorithm the minimum delay round-trip routes are selected without considering the route cost. A route thus selected is rejected if its delay is larger than the specified constraint.

MinCost: This is the algorithm proposed in the current work. That is, the round-trip routes are selected so that they are of minimum cost among those that they satisfy the delay constraint.

Once the round-trip routes from any client node i to any node server j are selected using either the MinDelay or MinCost approach, the parameters c_{ij} are determined. We then employ the proposed algorithm to find a solution to Problem 3. For the simulations we used the pseudopolynomial algorithm since it works sufficiently well for the selected instances and its implementation is considerably simpler than the polynomial algorithm.

Table 6.2: Average total cost for different delay constraints.

D	100	200	400	500	600	800
MinCost	178638	127591	118069	115734	114504	113878
MinDelay	197832	188928	182178	182178	182178	182178

In Table 6.2 we present the average total cost of using the servers and utilizing the link bandwidth. We make the following observations. The cost for both algorithms decreases as the delay constraint increases and levels off after certain value of the delay constraint. This is explained as follows. For smaller delay constraints, several locations are becoming unreachable by the nodes. Hence the options of directing the node load to the various locations are reduced and as result the overall cost of the solution increases. The leveling-off of the computed cost is due to the fact that as the delay constraints become looser, all the minimum cost round-trip routes are selected by algorithm MinCost and all the minimum delay round-trip routes by algorithm MinDelay. We also observe in Table 6.2 that the total cost of algorithm MinCost is always smaller than MinDelay, as expected, and the significance is becoming more pronounced for larger delays. This behavior is again due to the manner in which routes are selected by the two algorithms for a given delay constraint. For strict delay constraints, both algorithms choose mainly the permissible minimum delay round-trip routes and hence they have similar performance. For looser constraints, the fact that MinCost picks the minimum cost round-trip routes that satisfy the delay constraint instead of simply the minimum delay route (as MinDelay does) allows it to reduce the routing cost.

Finally experiments were run for networks of various sizes in order to assert the performance of the proposed algorithm in terms of running time. We generated power law directed networks for $|V| = \{20, 50, 100, 150\}$, ratio $r = |E|/|V|$ equal to 3 and two delay constraints $D = \{100, 500\}$. For each $|V|$ and r we generated ten different networks and for each experiment the link cost and delay, the server types, as well as the load of the nodes were generated according to the methods previously described.

The experiments were run on a Pentium PC IV, 1.7GHz, 785MB RAM. In Figure 6.3 we present the average running time (in seconds) of the proposed algorithm. We make the following observations. For a fixed number of nodes and edges we observe a small increase of the running time due to the delay constraint. This can be explained by the increase of iterations induced by the algorithm responsible for the evaluation of the minimum cost round trip routes (Algorithm 3). For fixed delay constraint the running time of the algorithm increases significantly when the number of nodes and edges increases. We also run experiments with a delay constraint equal to 800. The performance of the algorithm was similar with the performance results of $D = 500$ and therefore are not presented here. It is worth noting that the performance of the algorithm depends mainly on the number of nodes and edges and not on delay constraints. Although pseudopolynomial, tests with a wide variety of networks show that the algorithm has fairly satisfactory performance. Note that we have assumed $H = V$, that is, servers may be placed in

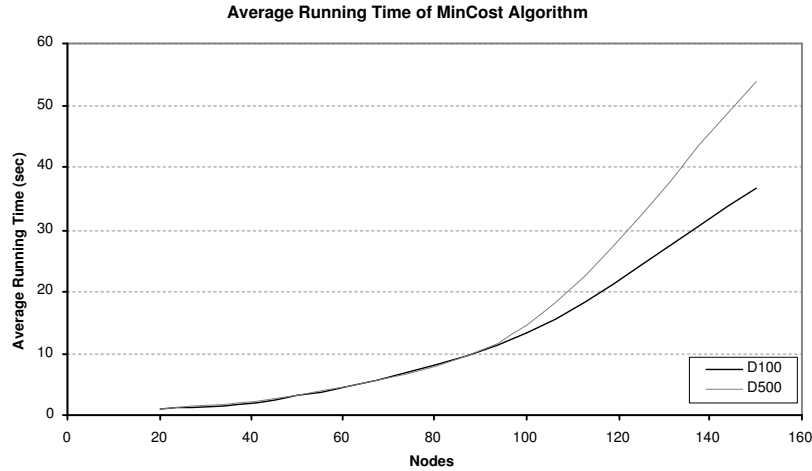


Figure 6.3: Average running time in power law networks, $D = \{100, 500\}$.

any of the nodes. In practical systems servers are placed in specific locations, that is $H \subset V$. In such cases the running time of the algorithm must decrease. Indeed, we run an experiment for a network with $|V| = 100$ nodes, ratio equal to 3, constraint $D = 500$ and ten possible server locations, that is $|H| = 10$, which are randomly chosen among the set of nodes V . The running time and the overall cost were found equal to 3.5 sec and 269722 respectively. By assuming $H = V$ and repeating the latter experiment we observe a) an increase of the running time of 15.9sec, b) a decrease of the overall cost of 120467 and finally the placement of servers in 17 different nodes.

6.5 Conclusion

We presented a pseudopolynomial approximation algorithm and a polynomial time algorithm for the NP-hard problem of replicated server placement with QoS constraints, in a general network context. The pseudopolynomial algorithm works well in several practical instances and is simpler than the polynomial time algorithm. The polynomial time algorithm is significant from a theoretical point of view and can be useful to employ if the problem instance renders the pseudopolynomial time algorithm very slow.

In this work we did not consider link capacities. It is an interesting open problem to incorporate the latter constraint into the problem. Another problem of interest is to consider the case where not all the database is replicated to each of the servers. Finally, it would be possible to adapt our algorithms specifically in order to provide proxy-based multicast services, taking also into consideration QoS guarantees.

Appendix: Proofs of the Lemmas

Running Time of Algorithm 4

In this section we show how the average time of Algorithm 4 can be improved, we present the worst case running time of Algorithm 4 and provide proofs of the lemmas.

Step Function $f_j(y)$

In this section it is shown that the average running time of Algorithm 4 can be improved by taking advantage of the fact that in this case $f_j(y)$ is a step function. For simplicity we drop the index j . Assume the unassigned nodes arranged in non-decreasing order of their costs $c_1 \leq c_2 \leq \dots \leq c_m$. Let l be the load already assigned. We then have to compute

$$\min_k \frac{f(l+k) - f(l) + \sum_{i=1}^k c_i y_i + y c_{k+1}}{k}.$$

Let

$$Y_k = \sum_{i=1}^k y_i, \quad C_k = \sum_{i=1}^k c_i y_i, \quad k = Y_k + y, \quad 0 \leq y \leq g_{k+1},$$

and

$$\begin{aligned} t(y) &= \frac{f(l+k) - f(l) + \sum_{i=1}^k c_i y_i + y c_{k+1}}{k} = \\ &= \frac{f(l + Y_k + y) - f(l) + C_k + c_{k+1} y}{Y_k + y} = \\ &= \frac{f(l + Y_k + y) - f(l) + C_k - c_{k+1} Y_k}{Y_k + y} + c_{k+1}. \end{aligned} \quad (6.22)$$

Assume that $f(z)$ is a step function

$$f(z) = f_m, \text{ if } z_m \leq z < z_{m+1}, \quad m = 1, 2, \dots,$$

where $z_0 = 0$, $f(0) = 0$. We call z_m the point of the m^{th} jump of the function $f(z)$.

Assume that $y \geq 0$ can take maximum value y_{\max}^{k+1} , i.e., y_{\max}^{k+1} is the maximum remaining load of the node connected through link $k+1$.

For the rest of the discussion, Figure 6.4 will be helpful

Let m_{k+1} be the index of the last point before $l + Y_k$ at which a jump of the function $f(z)$ occurs,

$$m_{k+1} = \max \{m : z_m \leq l + Y_k\}.$$

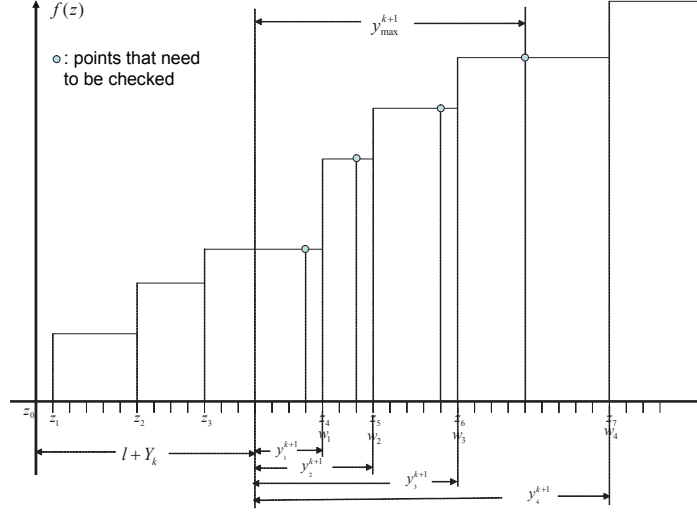


Figure 6.4: Points that need to be considered when checking c_{k+1} .

Also, let w_m^{k+1} be the m^{th} point after $l + Y_k$ at which a jump of $f(z)$ occurs,

$$w_m^{k+1} = z_{m_{k+1}+m}, \quad m = 1, 2, \dots,$$

and

$$y_m^{k+1} = w_m^{k+1} - l - Y_k.$$

Observe from (6.22) the following:

If

$$f(l + Y_k) - f(l) + C_k - c_{k+1}Y_k < 0,$$

then $t(y) < c_{k+1}$ for any y , $1 \leq y \leq y_1^{k+1}$ and the algorithm must stop. Note that this cannot happen when we examine c_1 and hence the algorithm always gives some output before it stops.

If on the other hand

$$f(l + Y_k) - f(l) + C_k - c_{k+1}Y_k \geq 0,$$

then the minimum is achieved at $\min \{y_1^{k+1} - 1, y_{\max}\}$. Moreover, in this case it holds (since $f(z)$ is increasing),

$$f(l + Y_k + y_m^{k+1}) - f(l) + C_k - c_{k+1}Y_k \geq 0, \quad \text{for all } m,$$

which implies that the minimum in the interval $[y_m^{k+1}, y_{m+1}^{k+1} - 1]$ is achieved at $\min \{y_{m+1}^{k+1} - 1, y_{\max}\}$.

The above discussion implies that when we need to find the minimum value of $t(y)$, we need only check its values at the points

$$y = y_m^{k+1} - 1, \text{ where } y_m^{k+1} - 1 \leq y_{\max}, \text{ and } y = y_{\max}.$$

Running time

We discuss below an implementation of Algorithm 4 and present its complexity. Let $|V|$ and $|H|$ be the number of client and server locations respectively. We assume for simplicity that $g_{\max} = 1$. As discussed in Section 6.2.3, the general case can be treated by considering a modified network with at most $|V|g_{\max}$ nodes. We also assume unit cost for compute $f_j(k)$. All references to steps below, concern Algorithm 4.

The implementation is the following.

1. We construct $|H|$ doubly linked lists, one per server location, where each list, L_j , $j \in H$, holds c_{ij} , for all $i \in V$ that are connected to j , in increasing order. Since each sorting takes time $O(|V| \log |V|)$, the time to construct the $|H|$ linked lists is $O(|H| |V| \log |V|)$.
2. We also create a matrix $M(i, j)$, $i \in V$, $j \in H$, where $M(i, j)$ holds the address of link (i, j) in the list L_j . If link (i, j) does not exist in L_j , then we set $M(i, j)$ to null. This takes time $O(|H| |V|)$. The matrix $M(i, j)$ is used to erase efficiently an element from L_j (see line 3b).
3. In the while loop (step 3) the following operations are performed.
 - (a) The $t(j)$ for each server location $j \in H$ are computed (step 5). For this, we need to scan through the list L_j , which takes $O(|V|)$ time (since each L_j contains at most $|V|$ elements). Hence to compute all $t(j)$, $j \in H$ and the minimum of these $t(j)$ (step 7), takes time $O(|H| |V|)$.
 - (b) The updating of server and client location loads (steps 9-13) takes $O(1)$ time. However, whenever the remaining load of a node becomes 0 (step 11), the node must be removed from further consideration. Hence, it must be removed from all linked lists L_j , $j \in H$. Therefore, for each of the k^* client locations, say location i , we check first if the link (i, j) exists in the list L_j , *i.e.*, check whether $M(i, j)$ is non null. If so, $M(i, j)$ contains the address of link (i, j) in L_j ; using this address we remove link (i, j) from L_j and we set $M(i, j)$ to null. This takes time $O(k^* |H|)$. Note that we can dispense with the matrix $M(i, j)$ altogether and search through the whole list L_j in order to remove the link (i, j) . This will increase the complexity of this operation to $O(k^* |V| |H|)$ but will also eliminate the space needed to store the matrix $M(i, j)$. The overall complexity of the algorithm will not be affected.

Regarding the overall complexity, we have the following.

- Line 1 is executed outside the while loop and hence takes time $O(|H| |V| \log |V|)$.
- Line 3a is executed inside the while loop. Since in the worst case only one client location may be removed at each iteration, this step may be executed at most $|V|$ times. Hence the total complexity of this operation is $O(|H| |V|^2)$.
- If k_n is the number of customers allocated to a facility at iteration n , and it takes m iterations for the algorithm to end, we have $k_1 + k_2 + k_3 + \dots + k_m = |V|$. Hence line 3b is executed in time

$$O(k_1 |H|) + O(k_2 |H|) + \dots + O(k_m |H|) = O(|H| |V|).$$

From the above we conclude that the whole algorithm executes in worst case time,

$$O\left(|H| |V| \log(|V|) + |H| |V|^2\right) = O\left(|H| |V|^2\right),$$

which is dominated by the time to compute $t(j)$ in step 5.

For general loads, replacing $|V|$ with $|V| g_{\max}$, the complexity becomes $O\left(|H| |V|^2 g_{\max}^2\right)$. Since in our setup $|H|$ can be as large as $|V|$, the overall complexity of the algorithm is $O(|V|^3 g_{\max}^2)$.

Proof of Lemma 1

Lemma 1: Consider the problems

$$\min_{\mathbf{x} \in A} g(\mathbf{x}), \quad A \subseteq R^n, \quad (6.23)$$

$$\min_{\mathbf{x} \in A} \bar{g}(\mathbf{x}), \quad A \subseteq R^n, \quad (6.24)$$

where $g(\mathbf{x}) \geq 0$. If for $\mathbf{x} \in A$,

$$g(\mathbf{x}) \leq \bar{g}(\mathbf{x}) \leq b g(\mathbf{x}), \quad (6.25)$$

then an a -approximate solution for problem (6.24), $a \geq 1$, is a ba -approximate solution for (6.23).

Proof: Let x^* , \bar{x}^* be the optimal solutions to (6.23), (6.24) respectively. Let also $\bar{\mathbf{y}}$ be an a -approximate solution for problem (6.24), *i.e.*,

$$\bar{g}(\bar{\mathbf{y}}) \leq a \bar{g}(\bar{\mathbf{x}}^*). \quad (6.26)$$

Since \bar{x}^* is optimal for problem (6.24), it holds

$$\bar{g}(\bar{\mathbf{x}}^*) \leq \bar{g}(\mathbf{x}^*) \leq b g(\mathbf{x}^*) \quad \text{by (6.25)}. \quad (6.27)$$

Then

$$\begin{aligned}
g(\mathbf{x}^*) &\leq g(\bar{\mathbf{y}}) \text{ since } \mathbf{x}^* \text{ solves (6.23)} \\
&\leq \bar{g}(\bar{\mathbf{y}}) \quad \text{by (6.25)} \\
&\leq a\bar{g}(\bar{\mathbf{x}}^*) \quad \text{by (6.26)} \\
&\leq abg(\mathbf{x}^*) \quad \text{by (6.27)}.
\end{aligned}$$

Hence $\bar{\mathbf{y}}$ is a ab solution to (6.23).

Proof of Lemma 2

Lemma 2: If the functions $f_i(y)$ are concave and Algorithm 4 is applied, then the load of each node in V can be assigned to a single server.

Proof: Assume that at the beginning of the t th iteration of the while loop in Algorithm 4 (step 3) the load of any assigned node, has been actually assigned to a single server. We will show that the same is true at the beginning of the $t + 1$ th iteration. The result will follow by induction.

It suffices to show that the minimum in step 5 is achieved when all the load of each unassigned node is assigned to a single server. To simplify notation, let $c_1 \leq c_2 \leq \dots \leq c_m$ be the unassigned nodes connected to node j , and let l be the load assigned to node j at the t th iteration. For simplicity, we drop the index j from the notation. We then have to compute

$$\min_k \frac{f(l+k) - f(l) + \sum_{i=1}^{n(k)} c_i g_i + l(k)c_{n(k)+1}}{k}. \quad (6.28)$$

Let $Y = \sum_{i=1}^n g_i$ and $k = Y + y$, $0 \leq y \leq g_{n+1}$. It suffices to show that the minimum of

$$s(y) = \frac{f(l+Y+y) - f(l) + \sum_{i=1}^n c_i g_i + y c_{n+1}}{Y+y}, \quad 0 \leq y \leq g_{n+1},$$

is achieved either at $y = 0$ or at $y = g_{n+1}$. Indeed this implies that the minimum in (6.28) is achieved when $k = \sum_{i=1}^j g_i$ for some j , $1 \leq j \leq m$, *i.e.*, that all the node loads are assigned to a single server.

Let us rewrite

$$\begin{aligned}
s(y) &= \frac{f(l+Y+y) - f(l) + \sum_{i=1}^n c_i g_i + c_{n+1}y}{Y+y} \\
&= \frac{f(l+Y+y) - f(l) + \sum_{i=1}^n (c_i - c_{n+1}) g_i}{Y+y} + c_{n+1} \\
&= \frac{\phi(Y+y)}{Y+y} + c_{n+1},
\end{aligned}$$

where $\phi(x) = f(l+x) - f(l) + \sum_{i=1}^n (c_i - c_{n+1}) g_i$.

Notice that $\phi(x)$ is concave. Suppose that a minimum of $s(y)$ occurs at $y = y_0$, such that $0 < y_0 < g_{n+1}$. By the definition of y_0 and the fact that $y_0 > 0$, we have

$$\frac{\phi(Y + y_0)}{Y + y_0} \leq \frac{\phi(Y + y_0 - 1)}{Y + y_0 - 1}$$

or

$$\phi(Y + y_0) - \phi(Y + y_0 - 1) \leq \frac{\phi(Y + y_0)}{Y + y_0}.$$

Hence, using the concavity of $\phi(x)$,

$$\phi(Y + y_0 + 1) - \phi(Y + y_0) \leq \frac{\phi(Y + y_0)}{Y + y_0},$$

and

$$\frac{\phi(Y + y_0 + 1)}{Y + y_0 + 1} \leq \frac{\phi(Y + y_0)}{Y + y_0}.$$

Therefore $y_0 + 1$ is a minimum too. Similarly, using the fact that $y_0 < g_{n+1}$ we conclude that $y_0 - 1$ is also a minimum.

This implies by induction that $\frac{\phi(Y+y)}{Y+y}$ is constant for $0 \leq y \leq g_{n+1}$ and hence the same holds for $s(y)$. We conclude that either the minimum is achieved at $y = 0$ or at $y = g_{n+1}$, or $s(y)$ is a constant for $0 \leq y \leq g_{n+1}$. In the latter case we can pick $y = g_{n+1}$ as the minimizing point.

Proof of Lemma 3

Lemma 3: If a function $\phi : \{0, 1, \dots, W\} \rightarrow \mathbb{Q}$ is subadditive and non-decreasing, then it holds for its upper hull $\phi_2(y)$, $\phi_2(y) \leq 2\phi(y)$.

Proof: For $y = 0$ and $y = W$ we have by definition $\phi_2(y) = \phi(y)$. Let now y be an integer such that $y_k < y \leq y_{k+1}$. Consider two cases:

1. $y_{k+1} - y_k < y$. Then

$$\begin{aligned} \phi_2(y) &\leq \phi(y_{k+1}) \text{ by (6.15)} \\ &\leq \phi(y_k + y) \text{ since } \phi \text{ is non-decreasing} \\ &\leq \phi(y_k) + \phi(y) \text{ by subadditivity} \\ &\leq 2\phi(y) \text{ since } \phi \text{ is non-decreasing.} \end{aligned}$$

2. $y_{k+1} - y_k \geq y$. Since $\phi_2(y)$ is piecewise linear and y_k, y_{k+1} are consecutive break points, it holds for any z , $y_k \leq z < y_{k+1}$,

$$\frac{\phi_2(y_{k+1}) - \phi_2(y_k)}{y_{k+1} - y_k} = \frac{\phi_2(y_{k+1}) - \phi_2(z)}{y_{k+1} - z}.$$

Since $\phi_2(y_k) = \phi(y_k)$, $k = 1, \dots, K$,

$$\begin{aligned} \frac{\phi(y_{k+1}) - \phi(y_k)}{y_{k+1} - y_k} &= \frac{\phi(y_{k+1}) - \phi_2(z)}{y_{k+1} - z} \\ &\leq \frac{\phi(y_{k+1}) - \phi(z)}{y_{k+1} - z} \text{ by (6.16)} \\ &\leq \frac{\phi(y_{k+1} - z)}{y_{k+1} - z} \text{ by subadditivity.} \end{aligned} \quad (6.29)$$

Now set $z = y_{k+1} - y$. Since $y_k \leq z < y_{k+1}$ by hypothesis, using the last inequality we have,

$$\begin{aligned} \phi_2(y) &= \phi(y_k) + \frac{\phi(y_{k+1}) - \phi(y_k)}{y_{k+1} - y_k}(y - y_k) \\ &\leq \phi(y_k) + \frac{\phi(y)}{y}(y - y_k) \text{ by (6.29)} \\ &\leq \phi(y_k) + \phi(y) \\ &\leq 2\phi(y) \text{ since } \phi \text{ is non-decreasing.} \end{aligned}$$

This completes the proof.

Proof of Lemma 4

Lemma 4: a) In Algorithm 5, $K = O\left(\frac{1}{\epsilon} \log \frac{\bar{f}(W)}{\bar{f}_1}\right)$.

b) The worst case running time of Algorithm 5 is $O\left(T \log(W) \frac{1}{\epsilon} \log \frac{\bar{f}(W)}{\bar{f}_1}\right)$, where T is the worst case time (over all y , $1 \leq y \leq W$) needed to compute $\bar{f}(y)$.

c) Consider the step function defined as follows: if $y_k \leq y < y_{k+1}$ for some k , $1 \leq k \leq K - 1$, then $\hat{f}(y) = \hat{f}_k$, and $\hat{f}(W) = \hat{f}_K$. It holds,

$$\hat{f}(y) \leq \bar{f}(y) \leq (1 + \epsilon)\hat{f}(y). \quad (6.30)$$

Proof: a) According to steps 2 and 4 it holds,

$$\bar{f}(y_k - 1) \leq (1 + \epsilon_1)\bar{f}(y_{k-1}) \leq \bar{f}(y_k). \quad (6.31)$$

Since $\bar{f}(y)$ is non-decreasing and $\bar{f}(1) \geq 0$ (hence $\bar{f}(y) > 0$, $y \geq 1$), inequality (6.31) implies that $y_{k-1} < y_k$. Moreover, $(1 + \epsilon)\hat{f}_{k-1} \leq (1 + \epsilon)\bar{f}(y_{k-1}) = \hat{f}_k$. This implies that $(1 + \epsilon)^{k-1} \hat{f}_1 \leq \hat{f}_k$. Hence the algorithm stops in $O\left(\log_{1+\epsilon} \frac{\bar{f}(W)}{\bar{f}_1}\right) = O\left(\log \frac{\bar{f}(W)}{\bar{f}_1} / \epsilon\right)$ steps.

b) Since $\bar{f}(y)$ is non-decreasing, we can use binary search in $[1, W]$ to find each y_k . Hence the determination of each y_k takes $O(T \log W)$ computations and the total running time is $O(TK \log W)$.

c) For $y_k \leq y < y_{k+1}$, taking into account (6.31) we have

$$\begin{aligned}\bar{f}(y) &\leq \bar{f}(y_{k+1} - 1) \text{ since } \bar{f}(y) \text{ is non-decreasing} \\ &\leq (1 + \epsilon)\bar{f}(y_k) = (1 + \epsilon)\hat{f}(y).\end{aligned}$$

By definition, $\hat{f}(y) = \bar{f}(y_k)$. Taking also into account that $\bar{f}(y_k) \leq \bar{f}(y)$, (6.30) follows.

Proof of Lemma 5

Lemma 5: Let $\epsilon_0 > 0$, $\epsilon > 0$ be given. Let $f(y)$ be the optimal solution to Problem 2 and assume that we compute for a given y the approximate function $\bar{f}(y)$ so that

$$f(y) \leq \bar{f}(y) \leq (1 + \epsilon_0) f(y). \quad (6.32)$$

a) For the purposes of computing the step function $\hat{f}(y)$ satisfying (6.30), $\bar{f}(y)$ may be assumed non-decreasing.

b) It holds for $\tilde{f}(y) = (1 + \epsilon)\hat{f}(y)$

$$f(y) \leq \tilde{f}(y) \leq (1 + (\epsilon + \epsilon_0 + \epsilon\epsilon_0)) f(y). \quad (6.33)$$

c) The number of jump points of $\hat{f}(y)$, hence of $\tilde{f}(y)$, is $O\left(\frac{1}{\epsilon} \log(|V| g_{\max})\right)$ and the running time of Algorithm 5 is $O\left(\frac{1}{\epsilon} T (\log(|V| g_{\max}))^2\right)$, where T is the worst case time (over all y , $1 \leq y \leq W$) needed to compute $\bar{f}(y)$.

Proof: a) To see that for the purposes of constructing $\bar{f}(y)$ we can assume without loss of generality that $\bar{f}(y)$ is non-decreasing, proceed as follows. If at some point during the computations for the construction of $\bar{f}(y)$ we obtain for $y_1 < y_2$, $\bar{f}(y_1) > \bar{f}(y_2)$, then we can replace $\bar{f}(y_2)$ with $\bar{f}_1(y_2) = \bar{f}(y_1)$ without violating (6.32). Indeed,

$$\begin{aligned}f(y_2) &\leq \bar{f}(y_2) \text{ by (6.32)} \\ &< \bar{f}(y_1) = \bar{f}_1(y_2), \text{ by assumption,}\end{aligned}$$

and

$$\begin{aligned}\bar{f}_1(y_2) &= \bar{f}(y_1) \\ &\leq (1 + \epsilon_0) f(y_1) \text{ by (6.32)} \\ &\leq (1 + \epsilon_0) f(y_2) \text{ since } f(y) \text{ is increasing.}\end{aligned}$$

b) Let $y_k \leq y \leq y_{k+1} - 1$. We then have

$$\begin{aligned}f(y) &\leq \bar{f}(y) \text{ by (6.32)} \\ &\leq (1 + \epsilon)\hat{f}(y) \text{ by (6.30)} \\ &= \tilde{f}(y).\end{aligned} \quad (6.34)$$

On the other hand,

$$\begin{aligned} f(y) &\geq \frac{1}{1 + \epsilon_0} \bar{f}(y) \text{ by (6.32)} \\ &\geq \frac{1}{1 + \epsilon_0} \hat{f}(y) \text{ by (6.30)} \\ &= \frac{\tilde{f}(y)}{(1 + \epsilon_0)(1 + \epsilon)}. \end{aligned} \tag{6.35}$$

From (6.34), (6.35), we obtain (6.33).

c) Since $f(y)$ is subadditive, it holds $f(W)/f(1) \leq W$. The result then follows from Lemma 4 a) and b).

Conclusions and Perspectives

In this thesis, we were concerned with the analytical modeling and performance evaluation of large mobile ad hoc networks. We considered protocols of all layers starting from the MAC layer. In each case, we modeled and analyzed the studied protocols, with the ultimate goal to propose new algorithms or protocols to improve the system's performance. We presented simulations and numerical results to support our contributions, as well as a complete implementation in the case of the MOST protocol. We give a summary of our main results and their context in the following table.

<i>Chapter</i>	<i>Layer</i>	<i>Studied Protocols</i>	<i>New Results</i>	
			<i>Modeling/Analysis</i>	<i>Algorithms/Protocols</i>
2	2	IEEE 802.11 <i>interaction with:</i> OLSR	One-hop, multi-hop delay distribution in power law	<ul style="list-style-type: none"> – Asymptotically optimal delay based routing – Cross-layer delay estimation protocol
3	3	OLSR	<ul style="list-style-type: none"> – Neighborhood model in massive ad hoc networks – Protocol scalability 	Fish-Eye OLSR
4			<ul style="list-style-type: none"> – Multicast scaling properties – Multicast capacity 	<ul style="list-style-type: none"> – MOST protocol – Efficient MST algorithm
5	4	TCP <i>interaction with:</i> Ethernet, 802.11	<ul style="list-style-type: none"> – Throughput analysis – Traffic autocorrelations: long-term dependencies due to heavy-tailed delays 	
6	5	Internet services <i>interaction with:</i> QoS information from lower layers		Pseudopolynomial, polynomial algorithms for replicated server placement with QoS constraints

The models we considered involve a very large number of nodes and account for information theoretic considerations, such as the Gupta and Kumar result. Throughout the thesis, we took the approach of evaluating the protocols performance in an asymptotic setting. Interestingly enough, in numerous cases, the asymptotic considerations permit the polynomial treatment of otherwise NP-hard problems. Consequently, we were able to provide asymptotically optimal optimizations in Chapters 2, 3 and 4.

In the chapters examining the higher (transport and application) layers, our results can be considered in a more general setting, and can be applied in wired networks as well. However, in these cases, we commented on the relevance and possible implications of previously derived results, concerning lower layer protocols, in a wireless environment. Therefore, we obtained more complete models involving the entire system's behavior.

As possible directions for future work concerning the topics studied in this thesis, we outline here some ideas already discussed in the chapter conclusions:

- to combine the delay routing protocol with a mechanism providing dynamic delay control and admission control, in order to account for the impact of new connections in the network,
- to evaluate the performance of Fish Eye OLSR with mobility,
- to provide real network measurements concerning the performance of MOST,
- to characterize the autocorrelations of TCP traffic in different time scales, and thus to explain the multi-fractal nature of Internet traffic in particular cases,
- to generalize the replicated server placement algorithms in order to include link capacities, partially replicated databases and multicast services.

Additionally, we discuss some more general directions for further research:

- It would be interesting to establish a link between bandwidth and delay quality of service requirements in the context of wireless multi-hop networks. This would permit an adaptation of bandwidth-QoS routing solutions for delay sensitive applications as well. Such a link can be established using the notion of *equivalent bandwidth*. This notion refers to the effective bandwidth satisfying a given QoS constraint, such as the over-delay ratio which we considered in Chapter 2. In this particular case the equivalent bandwidth on a given path would correspond to the capacity of the network in transferring packets which satisfy this delay constraint. The delay distribution analysis we provided in the context of multi-hop IEEE 802.11 networks, actually allows the complete characterization of this capacity.
- In the same context, it would be beneficial to provide a more complete model of the performance of the TCP protocol, in conjunction with the delay and bandwidth analyses of the IEEE 802.11 MAC. As we saw, the delay variations resulting from

the analysis of the MAC mechanisms are detrimental to TCP performance. In fact, significant efforts have been devoted in the research community to adapt the TCP protocol to wireless multi-hop networks, however there is still a need for detailed analytical models considering the interaction of the different protocols with regards to the system's performance. Based on these models and the equivalent bandwidth calculations, it would be possible for instance to optimize routing with regards to TCP performance.

- Finally, an important topic is the introduction of QoS mechanisms for multicast communications in wireless networks. The MOST protocol which is based on overlay multicast trees and unicast tunnels provides an excellent basis for this goal. The next step would be to incorporate QoS routing possibilities, and to adapt unicast admission control mechanisms for the construction of the overlay multicast trees. This is an interesting and challenging problem from an algorithmic, as well as a protocol design point of view.

List of Figures

1.1	Layers and protocols considered in the context of this work.	6
1.2	Generic OLSR packet format.	11
1.3	Diffusion of a message using pure flooding and MPR flooding.	12
2.1	Example of medium occupancy in a successful unicast transmission using the DCF channel access mechanism.	18
2.2	Binary exponential back-off.	18
2.3	Coefficients of $\beta(z)$	21
2.4	Coefficients of $w(z)$	22
2.5	Topology 1.	28
2.6	Analytic service time distribution.	28
2.7	Measured service time distribution.	29
2.8	Measured node delay distribution.	29
2.9	Topology 2.	30
2.10	Comparison between end-to-end delay distribution and the corresponding power law.	31
2.11	Comparison between the end-to-end distribution and the convolution of single hop distributions.	32
2.12	Delay estimation protocol framework.	34
3.1	Quantity $P(W < x)$ versus x for $\alpha = 2.5$, no fading.	43
3.2	Quantity $p_0 r$ versus r for $\alpha = 2.5$, no fading.	44
3.3	Reception range r and quantity $p_0 r$ versus p_0 for $\alpha = 2.5$, no fading.	44
3.4	Optimal threshold p_0 versus α	45

3.5	Average MPR set of a node versus neighborhood size.	47
3.6	Neighbourhood size versus the network size, $\alpha = 2.5$, no fading, respectively for F-OLSR (bottom) and OLSR (top).	49
3.7	Example of function ϕ used for Fish Eye strategy.	50
3.8	Neighborhood size versus the network size, $\alpha = 2.5$, no fading, respectively for OLSR (bottom) and OLSR with Fish Eye (top).	51
3.9	Maximum overall capacity versus the network size, $\alpha = 2.5$, no fading, respectively for OLSR (bottom) and OLSR with Fish eye (top).	52
4.1	Comparison of a Minimum Spanning Tree (MST) with an optimal Steiner tree in an Euclidean graph.	58
4.2	Multicast cost $R(n)$ versus multicast group size n	62
4.3	Shortest path tree cost $R(n)$ versus multicast group size n	63
4.4	Total number of multicast retransmissions versus group size n , compared with the number of retransmissions using MPR flooding (straight lines).	64
4.5	Total number of multicast retransmissions versus group size n , compared with the number of retransmissions using MPR flooding in a denser network (with 1025 nodes).	65
4.6	Join message content format.	70
4.7	Overview of multicast implementation.	72
4.8	Comparison of multicast versus unicast to all destinations.	73
4.9	Simulation results vs theoretical upper bound.	74
4.10	Delivery ratio vs group size for different source rates.	75
4.11	Delivery ratio (%) vs number of groups in the network.	75
4.12	Delivery ratio (%) vs group size with different mobility speeds.	76
4.13	Duplicate traffic load (%) vs group size with different mobility speeds.	76
5.1	Buffer filled with incoming traffic accessing a MAC channel.	81
5.2	Linear cumulative throughput evolution versus time n when $B = 1.5$	85
5.3	Cumulative throughput in $\frac{n}{\log^2 n}$ versus time n in the limit case of $B = 1$	85
5.4	Cumulative throughput in n^B versus time n when $B = 0.5$	86
5.5	Aggregated variance in m^{1-B} versus measure interval m , when $B = 1.5$	89

5.6	Aggregated variance in $\frac{1}{\log^2 m}$ versus measure interval m , in the limit case of $B = 1$	89
5.7	Aggregated variance in m^{-B} versus measure interval m , when $B = 0.5$	90
5.8	N TCP connections towards the same bottleneck buffer.	92
5.9	Limiting function $g(x)$ of the window size distribution.	93
5.10	Markovian TCP model.	94
5.11	Spectral gap of the TCP Markov chain for different error rates a	97
5.12	TCP traffic autocorrelations for error rates (a) $a = 0.05$, (b) $a = 0.005$. The time unit is the RTT.	97
5.13	Several TCP connections sharing the same link.	98
5.14	Traffic autocorrelations of 50 TCP connections with heavy tailed RTT distributions (a) $D_i = 40i^{0.8}$, (b) $D_i = 40i$. The time unit is 500ms.	100
6.1	a) A graph with five clients, $V = \{s, u, v, w, i\}$, two possible locations $H = \{s, j\}$ and two types of servers for each location. b) The modified graph where each link represents the round-trip minimum cost routes, satisfying the delay constraint between a client and a possible location. Cost functions $f_s(s)$ and $f_j(y)$ of the possible locations. c) The resulting graph. Servers have been placed at the appropriate locations.	109
6.2	A subadditive step function $\phi(y)$ with its jump points y_1, \dots, y_6 and its upper hull $\phi_2(y)$	118
6.3	Average running time in power law networks, $D = \{100, 500\}$	124
6.4	Points that need to be considered when checking c_{k+1}	126

List of Tables

2.1	Delay requirements for voice traffic.	15
2.2	Simulation Settings.	27
2.3	Measured parameters.	28
2.4	Collision probabilities for each hop of the path.	30
4.1	Common simulation parameters.	72
6.1	Notation table.	108
6.2	Average total cost for different delay constraints.	123

Bibliography

Publications

Journals

- [1] C. Adjih, E. Baccelli, T. Clausen, P. Jacquet, and G. Rodolakis. Fish eye OLSR scaling properties. *IEEE Journal of Communication and Networks (JCN), Special Issue on Mobile Ad Hoc Wireless Networks*, 2004.
- [2] G. Rodolakis, S. Siachalou, and L. Georgiadis. Replicated server placement with QoS constraints. *IEEE Transactions on Parallel and Distributed Systems*, 17(10), 2006.

Conferences

- [3] P. Jacquet, A. Meraihi Naimi, and G. Rodolakis. Routing on asymptotic delays in IEEE 802.11 wireless ad hoc networks. In *First Workshop on Resource Allocation in Wireless NETWORKS*, 2005.
- [4] P. Jacquet and G. Rodolakis. Multicast scaling properties in massively dense ad hoc networks. In *SANSO*, 2005.
- [5] G. Rodolakis and P. Jacquet. An analytical evaluation of autocorrelations in TCP traffic. In *AINTEC*, 2005.
- [6] G. Rodolakis, S. Siachalou, and L. Georgiadis. Replicated server placement with QoS constraints. In *3rd International Workshop on QoS in Multiservice IP Networks, Catania, Italy*, 2005.

Reports

- [7] C. Adjih, P. Jacquet, G. Rodolakis, and N. Vvedenskaya. Performance of multiple TCP flows: an analytical approach, INRIA research report RR-5417. 2004.

- [8] P. Jacquet, A. Meraihi Naimi, and G. Rodolakis. Asymptotic delay analysis for cross-layer delay based routing in ad hoc networks, under submission.
- [9] G. Rodolakis, A. Meraihi Naimi, and A. Laouti. Multicast overlay spanning tree protocol for ad hoc networks, under submission.
- [10] G. Rodolakis, S. Siachalou, and L. Georgiadis. Replicated server placement with QoS constraints. Technical Report, <http://users.auth.gr/leonid/public/TechReplicated.pdf>.

References

- [11] Bluetooth Specifications, <http://www.bluetooth.com/dev/specifications.asp>.
- [12] MDFFP, <http://hipercom.inria.fr/smolsr-molsr/>.
- [13] OOLSR, <http://hipercom.inria.fr/oolsr/>.
- [14] IEEE 802.11 Standard, Wireless LAN Medium Access Control and Physical layer Specifications, 1997.
- [15] IEEE 802.16 Standard, Recommended Practice for Local and Metropolitan Area Networks: Coexistence of Fixed Broadband Wireless Access Systems, 2004.
- [16] IEEE 802.16e Standard, Recommended Practice for Local and Metropolitan Area Networks – Part 16: Air Interface for Fixed Broadband Wireless Access Systems, 2005.
- [17] C. Adjih, L. Georgiadis, P. Jacquet, and W. Szpankowski. Is the Internet fractal: The multicast power law revisited. In *SODA*, 2002.
- [18] C. Adjih, P. Jacquet, and L. Viennot. Computing connected dominating set with multipoint relays, INRIA research report RR-4597, 2002.
- [19] C. Adjih, P. Jacquet, and N. Vvedenskaya. Performance evaluation of a single queue under multi-user TCP/IP version 2, INRIA research report RR-4478, 2002.
- [20] M. Allman, V. Paxson, and W. Stevens. TCP congestion control, RFC 2581, 1999.
- [21] R. Andonov and S. Rajopadhye. A sparse knapsack algo-tech-cuit and its synthesis. In *International Conference on Application Specific Array Processors ASPA '94*. IEEE, 1994.
- [22] S. Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *Journal of the ACM*, 45(5):753–782, 1998.

- [23] V. Arya, N. Garg, R. Khandekar, K. Munagala, and V. Pandit. Local search heuristics for k-median and facility location problems. In *In Proceedings of the 33rd ACM Symposium on Theory of Computing*, pages 21–29, 2001.
- [24] F. Baccelli, K. Tchoumatchenko, and S. Zuyev. Markov paths on the Poisson-Delaunay graph with applications to routing in mobile networks. *Adv. Appl. Probab.*, 32(1):1–18, 2000.
- [25] H. Badis and K. Al Agha. Quality of service for ad hoc optimized link state routing protocol (QOLSR), internet draft, 2006.
- [26] H. Badis, A. Munaretto, K. Al Agha, and G. Pujolle. Optimal path selection in a link state QoS routing protocol. In *IEEE VTC*, May 2004.
- [27] D. Bertsimas and G. Van Ryzin. An asymptotic determination of the minimum spanning tree and minimum matching constants in geometrical probability. *Operations Research Letters*, 9:223–231, 1990.
- [28] G. Bianchi. Performance analysis of the IEEE802.11 distributed coordination function. *IEEE journal on selected areas in Communications*, 18(3), 2000.
- [29] A. Broido, E. Basic, and K.C. Claffy. Invariance of the Internet RTT spectrum. global RTT analysis, <http://www.caida.org/broido/rtt/rtt.html>, 2002.
- [30] T. Bu and D. Towsley. On distinguishing between Internet power law topology generators. In *INFOCOM*, 2002.
- [31] Y. Chen, R. Katz, and J. Kubiawicz. Dynamic replica placement for scalable content delivery. In *First International Workshop on Peer-to-Peer Systems*, pages 306–318, 2002.
- [32] Y. Chen, R. Katz, and J. Kubiawicz. SCAN: A dynamic scalable and efficient content distribution network. In *First International Conference on Pervasive Computing*, 2002.
- [33] J. C.-I. Chuang and M. A. Sirbu. Pricing multicast communication: A cost-based approach. *Telecommunication Systems*, 17(3):281–297, 2001.
- [34] F. Chudak and D. Shmoys. Improved approximation algorithms for the uncapacitated facility location problem. *SIAM Journal on Computing*, 33(1):1–25, 2003.
- [35] F. Chudak and D. Williamson. Improved approximation algorithms for capacitated facility location problems. In *Integer Programming and Combinatorial Optimization*, 1999.
- [36] T. Clausen. Combining temporal and spatial partial topology for manet routing - merging OLSR and FSR. In *WPMC*, 2003.

- [37] T. Clausen and P. Jacquet (editors). Optimized link state routing protocol (OLSR). RFC 3626, October 2003. Network Working Group.
- [38] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks. ACM SIGCOMM, 2002.
- [39] C. Cordeiro, H. Gossain, and D. Agrawal. Multicast over wireless mobile ad hoc networks: present and future directions. *IEEE Network*, Vol. 17, Issue: 1, 2003.
- [40] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Second Edition*. MIT Press, 2001.
- [41] S. Corson and J. Macker. Mobile ad hoc networking (MANET): Routing protocol performance issues and evaluation considerations, RFC 2501, 1999.
- [42] M. de Berg, O. Schwarzkopf, M. Van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2000.
- [43] D. Z. Du and F. K. Hwang. A proof of Gilbert-Pollak's conjecture on the steiner ratio. *Algorithmica*, 45:121–135, 1992.
- [44] J. Postel (ed.). Transmission control protocol, RFC 793, 1981.
- [45] R. Braden (ed.). Requirements for Internet hosts - communication layers, RFC 1122, 1989.
- [46] K. Fall and K. Varadhan. The ns manual, http://www.isi.edu/nsnam/doc/ns_doc.pdf.
- [47] W. Fenner. Internet Group Management Protocol, Version 2, RFC 2236, 1997.
- [48] D.R. Figueiredo, B. Liu, V. Misra, and D. Towsley. On the autocorrelation structure of TCP traffic. 2002.
- [49] P. Flajolet, X. Gourdon, and P. Dumas. Mellin transforms and asymptotics: Harmonic sums. *Theoretical Computer Science*, 144:3–58, 1995.
- [50] P. Flajolet and A. M. Odlyzko. Singularity analysis of generating functions. *SIAM J. Discrete Math.*, 3:216–240, 1990.
- [51] P. Flajolet and R. Sedgewick. Analytic combinatorics, <http://algo.inria.fr/flajolet/publications/books.html>.
- [52] S. Floyd and T. Henderson. The NewReno modification to TCP's fast recovery algorithm, RFC 2582, 1999.
- [53] K. Fukuda, H. Takayasu, and M. Takayasu. Origin of critical behavior in ethernet traffic. *Physica A*, Elsevier, 2000.

- [54] J. Garcia-Luna-Aceves and E. L. Madruga. The core-assisted mesh protocol. *Journal on Selected Areas in Communications*, 17(8):1380–1294, August 1999.
- [55] M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the Theory on NP-Completeness*. W. H. FREEMAN AND COMPANY, 1979.
- [56] M. Gerla, X. Hong, and G. Pei. Fisheye state routing protocol (FSR) for ad hoc networks, internet draft (expired), 2002.
- [57] R. Guerin and A. Orda. QoS-based routing in networks with inaccurate information: Theory and algorithms. *IEEE/ACM Transactions on Networking*, 7(3):350–364, june 1999.
- [58] P. Gupta and P. R. Kumar. Capacity of wireless networks. *IEEE Transactions on Information Theory*, IT-46(2):388–404, 2000.
- [59] M. T. Hajiaghavi, M. Mahdian, and V. S. Mirrokni. The facility location problem with general cost functions. *Networks*, 42(1):42–47, 2003.
- [60] R. Hassin. Approximation schemes for the restricted shortest path problem. *Mathematics of Operations Research*, 17(1):36–42, 1992.
- [61] V. Jacobson. Congestion avoidance and control. In *ACM SIGCOMM*, 1988.
- [62] V. Jacobson, R. Braden, and D. Borman. TCP extensions for high performance, RFC 1323, 1992.
- [63] P. Jacquet. Éléments de théorie analytique de l’information, modélisation et évaluation de performances, INRIA research report RR-3505, 1998.
- [64] P. Jacquet. Geometry of information propagation in massively dense ad hoc networks. In *MobiHoc*, 2004.
- [65] P. Jacquet, A. Laouiti, P. Minet, and L. Viennot. Performance analysis of OLSR multipoint relay flooding in two ad hoc wireless network models. *RSRCP*, Special issue on Mobility and Internet, December 2001.
- [66] P. Jacquet, A. Laouiti, P. Minet, and L. Viennot. Performance evaluation of multipoint relaying in mobile ad hoc networks. In *Networking, Pisa*, 2002.
- [67] K. Jain and V. Vazirani. Approximation algorithms for the metric facility location and k-median problems using the primal-dual schema and the lagrangian relaxation. *Journal of the ACM*, 48:274–296, 2001.
- [68] S. Jamin, C. Jiu, A. Kurc, D. Raz, and Y. Shavitt. Constrained mirror placement on the Internet. In *IEEE INFOCOM*, pages 31–40, April 2001.
- [69] J. Jetcheva and D. B. Johnson. Adaptive demand-driven multicast routing in multi-hop wireless ad hoc networks. In *Proceedings of the Second Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001)*, 2001.

- [70] X. Jia, X. Hu, D. Li, W. Wu, and D. Du. Placement of web-server proxies with consideration of read and update operations on the Internet. *The Computer Journal*, 46(4), 2003.
- [71] X. Jia, D. Li, X. Hu, and D. Du. Optimal placement of web proxies for replicated web servers in the Internet. *The Computer Journal*, 44(5):329–339, 2001.
- [72] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer-Verlag, 2004.
- [73] M. Korupolu, C. Plaxton, and R. Rajaraman. Analysis of a local search heuristic for facility location problems. *Journal of Algorithms*, 37:146–188, 2000.
- [74] R. Krishnan, D. Raz, and Y. Shavitt. The cache location problem. *IEEE/ACM Transactions on Networking*, 8(5):568–582, October 2000.
- [75] A. Laouiti, P. Jacquet, P. Minet, L. Viennot, T. Clausen, and C. Adjih. Multicast Optimized Link State Routing, INRIA research report RR-4721, 2003.
- [76] A. Laouiti, A. Qayyum, and L. Viennot. Multipoint relaying: An efficient technique for flooding in mobile wireless networks. In *35th Annual Hawaii International Conference on System Sciences (HICSS'2001)*. IEEE Computer Society, 2001.
- [77] S. Lee, W. Su, and M. Gerla. On demand multicast routing protocol in multihop wireless mobile networks. *ACM/Baltzer Mobile Networks and Applications, special issue on Multipoint Communication in Wireless Mobile Networks*, 2000.
- [78] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Transactions on Networking*, 2(1):1–15, february 1994.
- [79] B. Li, M. Golin, G. Italiano, and X. Deng. On the optimal placement of web proxies in the Internet. In *IEEE INFOCOM*, 1999.
- [80] M. Liu, R. R. Talpade, A. McAuley, and E. Bommaiah. AMRoute: Adhoc Multicast Routing Protocol. *UMD TechReport 99-8*.
- [81] D. H. Lorenz and D. Raz. A simple efficient approximation scheme for the restricted shortest path problem. *Operations Research Letters*, 28(5):213–221, June 2001.
- [82] M. Mahdian, E. Markakis, A. Saberi, and V. Varizani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *Journal of the ACM*, 50(6):795–824, 2003.
- [83] A. Median, A. Lakhina, I. Matta, and J. Byers. *BRITE: Universal Topology Generation from a User's Perspective*. Computer Science Department Boston University, 2001.

- [84] R. Metcalfe and D. Boggs. Ethernet: Distributed packet switching for local computer networks. *Communications of the ACM*, 19(5):395–405, july 1976.
- [85] P. Van Mieghem, H. de Neve, and F. Kuipers. Hop-by-hop quality of service routing. *Computer Networks*, 37:407–423, 2001.
- [86] J. Moy. OSPF version 2, RFC 2328, 1998.
- [87] A. Meraihi Naimi. Délai et routage dans les réseaux ad hoc 802.11, Thèse de Doctorat - Université de Versailles Saint-Quentin-En-Yvelines, 2005.
- [88] D. Q. Nguyen and P. Minet. QoS support and OLSR routing in a mobile ad hoc network. In *ICN'06*.
- [89] M. Pal, I. Tardos, and T. Wexler. Facility location with nonuniform hard capacities. In *IEEE Symposium on Foundations of Computer Science*, page 329, October 14-17, 2001.
- [90] K. Park and W. Willinger. *Self-similar traffics*. Wiley, 2000.
- [91] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (AODV) routing, RFC 3561, 2003.
- [92] G. Phillips, S. Shenker, and H. Tangmunarunkit. Scaling of multicast trees: Comments on the Chuang-Sirbu scaling law. In *SIGCOMM*, pages 41–51, 1999.
- [93] L. Qiu, V. Padmanabhan, and G. Voelker. On the placement of web server replicas. In *IEEE INFOCOM*, pages 1587–1596, April 2001.
- [94] E. Royer and C. Perkins. Multicast Ad hoc On-Demand Distance Vector (MAODV) Routing, IETF, Internet Draft: draft-ietf-manet-maodv-00.txt, 2000.
- [95] H. Sallay and O. Festor. A highly distributed dynamic IP multicast accounting and management framework. In *IFIP/IEEE Eighth International Symposium on Integrated Network Management*, 2003.
- [96] M. Sayal, Y. Breitbart, P. Scheuermann, and R. Vingralek. Selection algorithms for replicated web servers. In *Workshop on Internet Server Performance, Madison, Wisconsin*, 1998.
- [97] S. Siachalou and L. Georgiadis. Efficient QoS routing. *Computer Networks*, 43:351–367, 2003.
- [98] M. Steele. Growth rates of euclidean minimal spanning trees with power weighted edges. *Annals of Probability*, 16:1767–1787, 1988.
- [99] A. S. Tanenbaum. *Computer Networks, Fourth Edition*. Prentice Hall, 2003.

-
- [100] X. Tang and S. T. Chanson. The minimal cost distribution tree problem of recursive expiration-based consistency management. *IEEE Transactions on Parallel and Distributed Systems*, 15(3):214–227, March 2004.
 - [101] X. Tang and S. T. Chanson. Minimal cost replication of dynamic web contents under flat update delivery. *IEEE Transactions on Parallel and Distributed Systems*, 15(5):431–439, May 2004.
 - [102] X. Tang and J. Xu. On replica placement for QoS-aware content distribution. In *IEEE INFOCOM*, 2004.
 - [103] O. Tickoo and B. Sikdar. Queueing analysis and delay mitigation in IEEE 802.11 random access MAC based wireless networks. In *IEEE INFOCOM*, 2004.
 - [104] S. Toumpis and A. J. Goldsmith. Performance bounds for large wireless networks with mobile nodes and multicast traffic. In *Inter. Workshop on Wireless Ad Hoc Networks, Oulu, Finland*, 2004.
 - [105] V. Vazirani. *Approximation Algorithms*. Springer-Verlag, 2001.
 - [106] J. E. Wieselthier, G. D. Nguyen, and A. Ephremides. On the construction of energy-efficient broadcast and multicast trees in wireless networks. In *INFOCOM*, pages 585–594, 2000.
 - [107] H. Zhai and Y. Fang. Performance of wireless LANs based on IEEE 802.11 MAC protocol. In *IEEE PIMRC*, 2003.