Helsinki University of Technology Networking Laboratory
Teknillinen korkeakoulu Tietoverkkolaboratorio
Espoo 2007 Report 2/2007

TRAFFIC ENGINEERING IN THE INTERNET: FROM TRAFFIC CHARACTERIZATION TO LOAD BALANCING AND PEER-TO-PEER FILE SHARING

Riikka Susitaival

Dissertation for the degree of Doctor of Science in Technology to be presented with due permission for public examination and debate in Auditorium S1 at Helsinki University of Technology (Espoo, Finland) on the 19th of September, 2007, at 12 o'clock noon.

Helsinki University of Technology
Department of Electrical and Communications Engineering
Networking Laboratory

Teknillinen korkeakoulu
Sähkö- ja tietoliikennetekniikan osasto
Tietoverkkolaboratorio

Distribution:
Helsinki University of Technology
Networking Laboratory
P.O.Box 3000
FIN-02015 TKK
Tel. +358-9-451 2461
Fax. +358-9-451 2474

ISBN 978-951-22-8830-4
ISSN 1458-0322

Picaset Oy
Espoo 2007

TEKNILLINEN KORKEAKOULU
TEKNISKA HÖGSKOLAN
HELSINKI UNIVERSITY OF TECHNOLOGY

HELSINKI UNIVERSITY OF TECHNOLOGY P.O.BOX 1000, FIN-02015 TKK http://www.tkk.fi/	ABSTRACT OF DOCTORAL DISSERTATION
Author: Riikka Susitaival	

Name of the dissertation
Traffic Engineering in the Internet: From Traffic Characterization to Load Balancing and Peer-to-Peer File Sharing

Date of manuscript 12th of March 2007	Date of the dissertation 19th of September 2007
☐ Monograph	☒ Article dissertation (summary + original articles)

Department	Department of Electrical and Communications Engineering
Laboratory	Networking Laboratory
Field of Research	Teletraffic theory
Opponent	Professor Phuoc Tran-Gia (University of Würzburg, Germany)
Supervisor	Professor Jorma Virtamo (Helsinki University of Technology)
Instructor	Ph.D Samuli Aalto (Helsinki University of Technology)

Abstract

Traffic engineering refers to the performance optimization of operational networks. On one hand, traffic offered between origin and destination nodes loads the network and on the other hand, this traffic has to be carried in the network in such a way that performance objectives are fulfilled. In this thesis, we study three different problem areas related to traffic engineering covering traffic characterization, load balancing and peer-to-peer (P2P) file sharing.

In the first part of this thesis we characterize the measured traffic on a link in the Finnish backbone network Funet. Traffic on the link is first considered as an aggregate and then split into origin-destination pairs based on the IP addresses of the packets. At a fine level of spatial aggregation we identify four typical OD pair representatives called "Normal", "Bursty", "Uniform" and "Periodic". In particular, we are interested in how the so-called moving IID Gaussian model fits together with the aggregate link data and the OD pair representatives.

The second part of the thesis considers load balancing in different types of networks. The idea of load balancing is to move traffic from congested links to other parts of the network in a well-controlled way. If the traffic demands are known, the load balancing can be formulated as an optimization problem. However, knowledge of traffic demands is often lacking. For that reason we propose an adaptive and distributed algorithm that gradually balances the load by making small changes in the traffic-splitting ratios on the basis of measured link loads. The application of the adaptive algorithm for both MPLS as well as for OSPF networks is considered. By numerical evaluation we find that the adaptive algorithm converges rapidly almost to the optimum. We also develop optimization algorithms that differentiate traffic classes in terms of mean delay. In this thesis, the differentiation is achieved by the use of both routing and WFQ scheduling. Finally, the load balancing of wireless multihop networks is considered by formulating a linear optimization problem for the joint optimization of both routing and scheduling of the network.

In the third part of the thesis we consider the population dynamics and performance of novel P2P file-sharing networks. First we study the dynamics of distributing a single chunk by a deterministic fluid model as well as by a more detailed Markov model, which makes evaluation of the lifetime of the system possible. Then we extend the Markov chain model to the case of two chunks and compare the performance of different chunk or peer selection policies in terms of the lifetime as well as the download time of the file. Finally, with a spatio-temporal model we assess how much selecting the nearest peer instead of a random one reduces the usage of the resources of the underlying network.

Keywords	traffic engineering, Internet measurements, traffic characterization, load balancing, MPLS, OSPF, peer-to-peer, file sharing		
ISBN (printed)	978-951-22-8830-4	ISBN (pdf)	978-951-22-8831-1
ISSN	1458-0322	Number of pages	96 pp.
Publisher	Networking Laboratory / Helsinki University of Technology		
Print distribution			
☒ The dissertation can be read at http://lib.tkk.fi/Diss/			

TEKNILLINEN KORKEAKOULU
TEKNISKA HÖGSKOLAN
HELSINKI UNIVERSITY OF TECHNOLOGY

TEKNILLINEN KORKEAKOULU PL 1000, FIN-02015 TKK http://www.tkk.fi/	VÄITÖSKIRJAN TIIVISTELMÄ
Tekijä: Riikka Susitaival	
Väitöskirjan nimi Liikenteenhallinta Internetissä: liikenteen karakterisoinnista kuormantasaukseen ja tiedostonjakoon vertaisverkoissa	
Käsikirjoituksen jättämispäivämäärä 12. maaliskuuta 2007	Väitöstilaisuuden ajankohta 19. syyskuuta 2007
☐ Monografia	☒ Yhdistelmäväitöskirja (yhteenveto + erillisartikkelit)

Osasto	Sähkö-ja tietoliikennetekniikan osasto
Laboratorio	Tietoverkkolaboratorio
Tutkimusala	Teleliikenneteoria
Vastaväittäjä	Professori Phuoc Tran-Gia (Würzburgin yliopisto, Saksa)
Työn valvoja	Professori Jorma Virtamo (Teknillinen korkeakoulu)
Työn ohjaaja	Fil.tri Samuli Aalto (Teknillinen korkeakoulu)

Tiivistelmä

Liikenteenhallinnalla voidaan optimoida tietoverkkojen suorituskykyä. Toisaalta tarjottu liikenne lähde- ja määränpääsolmujen välillä kuormittaa verkkoa ja toisaalta tämä liikenne tulisi kuljettaa verkossa siten, että suorituskykyvaatimukset täyttyvät. Tässä väitöskirjassa tutkitaan kolmea eri liikenteenhallintaan liittyvää tutkimusongelmaa, nimittäin liikenteen karakterisointia, kuormantasausta ja tiedoston jakoa vertaisverkoissa.

Työn ensimmäinen osa käsittelee suomalaisesta runkoverkosta Funetista mitatun liikenteen karakterisointia. Linkin liikenne ajatellaan ensiksi yhtenä kokonaisuutena ja sen jälkeen se jaetaan lähde- ja määränpääpareihin perustuen pakettien IP-osoitteisiin. Jaon ollessa hieno voidaan erottaa neljä eri tyyppistä liikenneparia, jotka ovat normaalinen, purskeinen, tasajakautunut ja jaksollinen. Työssä ollaan erityisesti kiinnostuneita siitä, kuinka niin sanottu liukuva gaussinen liikennemalli IID-oletuksin sopii linkkidataan ja edustaviin lähde-määränpääpareihin.

Työn toinen osa tarkastelee kuormantasausta erilaisissa verkoissa. Kuormantasauksen ideana on siirtää hallitulla tavalla liikennettä verkon kuormittuneilta linkeiltä ruuhkattomampiin osiin. Jos tarjotut liikenteet tiedetään, voidaan kuormantasaus muotoilla optimointiongelmana. Kuitenkin usein tieto liikenteistä puuttuu. Tästä syystä työssä kehitetään adaptiivinen ja hajautettu algoritmi, joka tasaa kuormaa tekemällä asteittaisia muutoksia liikenteenjakosuhteissa mitattuun linkkitietoon perustuen. Kehitetyn algoritmin soveltamista tarkastellaan sekä MPLS- että OSPF-verkoissa. Numeerisin esimerkein osoitetaan, että adaptiivinen algoritmi saavuttaa nopeasti optimaalisen liikenteenjaon. Työssä kehitetään myös optimointialgoritmeja, jotka eriyttävät liikenneluokkia viiveen suhteen. Eriytystä saavutetaan sekä reitityksellä että WFQ-skeduloinnilla. Lopuksi kuormantasausta tutkitaan langattomissa monihyppyverkoissa muotoilemalla optimointiongelma, joka yhtäaikaisesti optimoi sekä reitityksen että skeduloinnin verkossa.

Työn viimeisessä osassa tarkastellaan populaatiodynamiikkaa ja suorituskykyä uudentyyppisissä tiedostonjakoverkoissa. Aluksi tutkitaan yhden tiedostonpalan levitystä deterministisellä nestemallilla ja tämän jälkeen yksityiskohtaisemmalla Markov-mallilla. Jälkimmäinen malli mahdollistaa myös tiedostonjaon elinajan arvioinnin. Yhden palan Markov-malli laajennetaan käsittämään kaksi palaa, mikä mahdollistaa erilaisten vertaissolmujen valintapolitiikkojen vertailun sekä elinajan että latausviiveen suhteen. Lopuksi työssä tutkitaan aika-paikka-mallin avulla, kuinka verkon resurssien käyttöä voidaan vähentää politiikalla, jossa valitaan lähin vertaissolmu satunnaisen sijaan.

Avainsanat	liikenteenhallinta, mittaukset Internetissä, liikenteen karakterisointi, kuormantasaus, MPLS, OSPF, vertaisverkot, tiedostonjako		
ISBN (painettu)	978-951-22-8830-4	ISBN (pdf)	978-951-22-8831-1
ISSN	1458-0322	Sivumäärä	96 s.
Julkaisija	Tietoverkkolaboratorio / Teknillinen Korkeakoulu		
Painetun väitöskirjan jakelu			
☒ Luettavissa verkossa osoitteessa http://lib.tkk.fi/Diss/			

PREFACE

This dissertation is the result of research that began in 2002, when I started my Master's Thesis in the Networking Laboratory. The first research project, in which I was involved, was COST279, funded by Tekes. After that, I have worked in several projects, such as Ironet, Pannet, and ABI, funded by Tekes, and FIT and CLOWN, funded by the Academy of Finland. During the years 2004 and 2005 my research was mainly funded by the Graduate School on Networks for Information Society (GSNIS), for which I would like to thank the Department of Electrical and Communications Engineering. In addition, the personal scholarships received from the TES and Nokia foundations are gratefully acknowledged.

First, I wish to thank Prof. Jorma Virtamo for all the efforts he has made in supervising my thesis. I appreciate the opportunity to work with such a highly skilled and motivated supervisor. Second, I am very grateful to my instructor, Dr. Samuli Aalto, for the long-term research co-operation. Without his help and advice the thesis would never have been completed. I am also delighted with the joint research carried out together with my colleague Mr. Ilmari Juva. In addition, Mr. Markus Peuhkuri has been very helpful in providing measurement data.

During my years in the Networking Laboratory, I have had many great people as colleagues, even though some of them have already left the laboratory. The joyful times spent during coffee breaks, as well as at parties outside of office hours, has made my efforts to complete the thesis more comfortable.

My family has supported me greatly during my studies, which I appreciate very much. I would also like to thank my great friends from both Mikkeli and TKK for all the time we have spent together. Finally, I would like to thank Teemu for all his encouragement and love.

CONTENTS

LIST OF PUBLICATIONS

[1] Ilmari Juva, Riikka Susitaival, Markus Peuhkuri, and Samuli Aalto. Traffic characterization for traffic engineering purposes: Analysis of Funet data. In *Proceedings of NGI 2005*, pages 404–411, April 2005.

[2] Riikka Susitaival, Ilmari Juva, Markus Peuhkuri, and Samuli Aalto. Characteristics of origin-destination pair traffic in Funet. *Telecommunication Systems*, (33):67–88, 2006.

[3] Ilmari Juva, Riikka Susitaival, Markus Peuhkuri, and Samuli Aalto. Effects of spatial aggregation on the characteristics of origin-destination pair traffic in Funet. Technical report 1/2007, Networking Laboratory, Helsinki University of Technology, ISBN 978-951-22-8681-2, 2007.

[4] Riikka Susitaival, Samuli Aalto, and Jorma Virtamo. Adaptive load balancing using MPLS. In *Proceedings of Joint Conference 12th GI/ITG Conference on Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB) together with 3rd Polish-German Teletraffic Symposium (PGTS)*, pages 15 – 24, 2004.

[5] Riikka Susitaival and Samuli Aalto. Adaptive load balancing with OSPF. In *Proceedings of the Second International Conference on the Performance Modelling and Evaluation of Heterogeneous Networks (HET-NETs '04)*, pages P9/1–P9/10, 2004.

[6] Riikka Susitaival, Jorma Virtamo, and Samuli Aalto. Load balancing by MPLS in differentiated services networks. In *Proceedings of International Workshop Art-QoS*, pages 252–264, March 2003.

[7] Riikka Susitaival. Load balancing by joint optimization of routing and scheduling in wireless mesh networks. In *Proceedings of the 20th International Teletraffic Congress (ITC)*, pages 483–494, 2007.

[8] Riikka Susitaival, Samuli Aalto, and Jorma Virtamo. Analyzing the dynamics and resource usage of P2P file sharing systems by a spatio-temporal model. In *Proceedings of International Workshop on P2P for High Performance Computational Sciences (P2P-HPCS06) in conjunction with ICCS*, pages 420–427, May 2006.

[9] Riikka Susitaival and Samuli Aalto. Modelling the population dynamics and the file availability in a BitTorrent-like P2P system with decreasing peer arrival rate. In *Proceedings of International Workshop on Self-Organizing Systems (IWSOS)*, pages 34–48, Sep. 2006.

[10] Riikka Susitaival and Samuli Aalto. Analyzing the file availability and download time in a P2P file sharing system. In *Proceedings of NGI 2007*, pages 88–95, 2007.

1 INTRODUCTION

1.1 Background

During the last decade the Internet has expanded into a world-wide network connecting millions of hosts and users and providing services for everyone. Many emerging applications are bandwidth-intensive in their nature; the size of downloaded files including music and videos can be huge, from ten megabits to many gigabits. The efficient use of network resources is thus crucial for the survivability of the Internet. Traffic engineering (TE) covers a range of mechanisms for optimizing operational networks from the traffic perspective. The time scale in traffic engineering varies from the short-term network control to network planning over a longer time period.

In traditional IP networks the tools for traffic engineering are limited due to the lack of control of the routes used. One purpose of Multiprotocol Label Switching (MPLS) is to provide TE with more capabilities. Explicit routing allows the packets to be routed along pre-defined paths. As the traffic is split into multiple parallel paths a finer distribution over the network is obtained. With the pervasion of MPLS, interest in developing similar traffic engineering mechanisms in shortest-path networks has emerged again.

To develop efficient traffic engineering mechanisms, information about the traffic offered to the network is essential. Many proposed traffic engineering mechanisms assume that the traffic demands are known. However, measuring the end-to-end demands is not a straightforward task. Even the common characteristics of origin-destination pair traffic are not fully understood in the research community. If the traffic demand matrix cannot be measured or it changes unpredictably, adaptive traffic engineering mechanisms are necessary. One way to develop traffic-aware engineering mechanisms is to first measure the link loads of the network, which is quite easy using common management protocols such as SNMP, and then make appropriate control actions.

The core of the Internet network consists of national and regional Internet Service Providers (ISPs) that are connected by fixed high-bandwidth links providing a capacity of up to ten gigabits per second. At the other end, Wireless Mesh Networks (WMNs) are dynamic networks that are constructed from mesh routers and mesh clients connected by wireless links [AW05]. With multi-hop routing the mesh routers form a mesh backbone which makes possible cost-effective communication between the clients in the network as well as access to the Internet. Together with growing interest in the commercial exploitation of WMNs the performance optimization of those networks has become an important issue.

Finally, the emerging demand for exchanging very large files effectively over the Internet has made it necessary to develop specific networks running on top of the Internet but utilizing its basic architecture. These types of networks, called peer-to-peer (P2P) networks, connect a huge number of peers acting as both clients and servers for each other. The architecture of a P2P network is not fixed. Instead, there are multiple different types of

P2P networks or protocols. Performance optimization is a natural part of the development of these protocols. On the other hand, peering itself can be viewed as a traffic engineering solution for reducing the load of central servers.

1.2 Traffic engineering

Traffic engineering refers to the optimization of operational networks. The idea is to use existing resources in the most effective manner which is complementary to dimensioning, where one adds resources in order to meet the requirements for serving customers properly. On one hand, we have a network which has a limited capacity, on the other hand there are customers who offer traffic to the network and have to be served. Both capacity and traffic demands vary in the long term as operators build new infrastructure and customer demands change along with the availability of new services.

Another issue related to the management of IP networks is the development of Quality of Service (QoS) mechanisms, in which the idea is to serve applications with various traffic characteristics differently. Traffic engineering and QoS are closely related, but, in order to differentiate them, QoS controls how the resources are allocated to different users, whereas TE controls spatial load allocation and resource usage.

In the optimization of operational networks the most important target is the minimization of congestion. Congestion can be divided into two types, congestion in the case where resources are insufficient to carry all the traffic, and congestion in the case where resources are divided inefficiently so that a part of the network is over-utilized while another part of the network has unused bandwidth. The second type of congestion can be effectively alleviated using techniques provided by traffic engineering, such as load balancing. The objective can then be minimizing the maximum link utilization, for example. However, traffic engineering should be carried out in such a way that congestion can be managed cost-effectively.

The performance optimization of networks is actually a control problem. Traffic engineering should provide sufficient control in an adaptive feedback control system. The tasks of a controller consist of the modification of traffic management parameters, the modification of routing parameters, and modifications of resource attributes and constraints [AMA+99].

In Figure 1.1 we depict one framework, based on the model presented in [FRT02], for traffic engineering in the Internet. In this model, the topology and traffic demands derived from the operational network act as input parameters for traffic engineering. On the basis of these parameters an optimal way to carry traffic over the network is defined. "Routing model" refers to the underlying routing protocol, such as OSPF or MPLS. Optimal routing is used as a target in the optimization of the routing parameters of the current routing protocol. For example, OSPF weights or explicit paths in MPLS can be determined on the basis of the optimal traffic allocation. In the optimization process, it is essential to have an exact view of the routing model, that is, how traffic is distributed over the network with given parameters. On the other hand, the routing model sets constraints for optimal traffic allocation (use of the shortest paths, for example). As a result, we

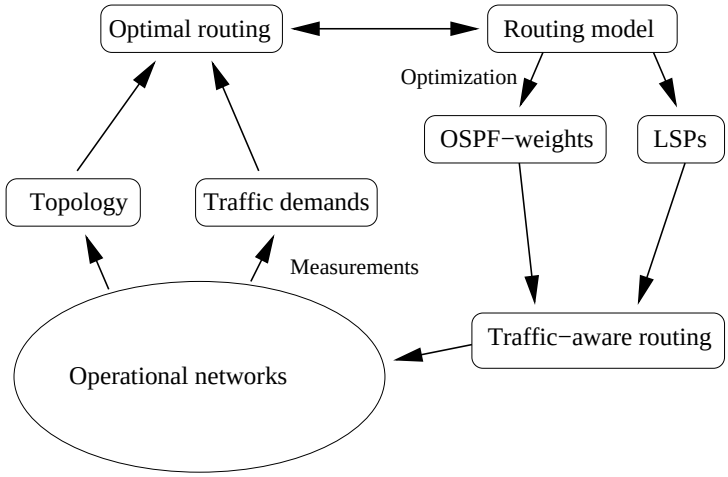

Figure 1.1: Traffic engineering model

have traffic-aware routing, which should provide a better performance than the standard shortest-path routing.

In the framework outlined above, traffic-aware routing is the only traffic engineering mechanism in use. Another mechanism to be considered in traffic engineering is admission control. However, this mechanism is beyond of the scope of this thesis.

Different functionalities of traffic engineering can be categorized on the basis of the time scale over which they operate. These time scales are traffic management, capacity management, and network planning. The response time of traffic management is from seconds to some minutes. The purpose of traffic management is to react to short-term congestion produced by unpredictable traffic fluctuations, changes in traffic demands, or to link failures. Capacity management consists of functionalities such as capacity planning and routing design on a time scale of days to weeks. Finally, network planning is responsible for predicting the long-term demand and dimensioning and deployment of the nodes and links on a time scale of months and years.

1.3 Adaptivity in network management

Some of the functions of the Internet are already adaptive. For example, TCP sources adjust their sending rate on the basis of the number of lost packets, and the flooding mechanism of the OSPF protocol informs nodes about the state of the links. However, these mechanisms do not guarantee that the usage of resources is effective network-wide.

In the days of the ARPANET there were already attempts to optimize the use of resources dynamically by changing the link metrics, which define the paths for the incoming traffic to the routers. As an example, the shortest paths were calculated using the measured link delays as the link costs. However, these attempts turned out to be unsuccessful, since changing the routes resulted in dramatic traffic oscillations. For a long time there

has been a general belief that it is too difficult to design a load-sensitive routing which is both sensitive and does not generate too much overhead.

In some cases where a large population of users can affect the routes used, adaptivity is a natural part of the system. If the user finds that the route currently being used is congested, it can select another one and thus the traffic load is gradually balanced. The Wardrop equilibria, defined in the context of transportation networks [War52], state that in the equilibrium state the journey times of the paths used between two end points are equal and shorter than the journey times of unused paths. However, the equilibrium does not imply that the stable solution obtained is optimal. Indeed, a counter-example is shown in [BKT97], where the addition of an extra link surprisingly diminishes the performance of a network.

There are many factors that have an influence on the success of adaptive algorithms. On one hand, technical aspects, such as the amount of overhead and cost of additional hard- and software, may restrict their wider use. On the other hand, the algorithms themselves have to satisfy certain requirements, such as stability and fast convergence to a good solution. If the control decisions are made on the basis of incomplete information on the state of the network, changes in the network parameters have to be made in small steps in order to avoid undesirable oscillations.

1.4 Routing in the Internet

In standard IP routing, the routers make independent decisions about how to forward packets to destinations. Each IPv4 datagram includes a 32-bit field for both the source and destination addresses. For example, in a class C address, the network part takes the first 24 bits (also known as the network prefix) and the host part 8 bits. When a packet arrives, the router forwards it to the next router according to the packet's destination address. The routers have to keep up a routing table that maps the destination address to the best next hop. Routing tables can be static or dynamic. Static routing tables are manually configured, whereas dynamic routing tables adapt to network changes.

Dynamic routing protocols can be divided into interior routing protocols and exterior routing protocols. The most common interior routing protocols are the Routing Information Protocol (RIP), Open Shortest Path First (OSPF), and Intermediate System-Intermediate System (IS-IS). RIP, as a distance vector protocol, uses distributed shortest path calculation, whereas link state routing protocols, such as OSPF and IS-IS, maintain the view of the topology of a local network in each node and, on the basis of that, calculate the best paths to all the destinations in the network. Traffic is routed along the shortest path, since the use of resources is then minimized. However, the routing protocols that are based on shortest path calculations do not take the congestion avoidance into account. Algorithms optimize the paths on the basis of simple metrics but the adequacy of the bandwidth is not considered.

Equal-Cost Multipath (ECMP) refers to a mechanism in which traffic is split equally into the parallel shortest paths from a router to a given destination [Moy98]. The support of ECMP improves the performance in

a situation where a traffic stream is routed over a link that does not have enough bandwidth. However, the equal-cost multipath algorithm does not help in cases where two different traffic streams are routed along the same congested link. In addition, the equal-cost multipath approach does not scale well to large networks.

The inadequacies of the routing protocols above can be overcome by overlay models such as IP over ATM and IP over Frame Relay. These models construct virtual topologies from virtual circuits (VC) and then provide services such as constraint-based routing at the VC level. Nevertheless, these techniques also have scalability problems.

Finally, MPLS (Multi Protocol Label Switching) is a flexible technology that makes possible new services in IP networks and makes routing more effective [RVC01]. In MPLS two different approaches, datagram and virtual circuit routing, are combined in a single technology. The idea of MPLS is to assign a label to the header of each packet at the ingress node of the MPLS network. After that the packet is forwarded over Label Switched Paths (LSPs) according to the label until it receives the egress node. Each label can be associated with a Forwarding Equivalence Class (FEC), which means a group of packets that can be treated in the same manner. The binding between a label and a FEC is distributed by a Label Distribution Protocol (LDP).

1.5 Peer-to-peer networks

Traditionally, the hosts of an IP network have been divided into clients and servers. The client programs of the end systems request and receive information from servers located in other end systems. Web browsing, e-mail, and file transfer and other well-known Internet applications are based on this kind of *client/server* architecture. However, this type of communication does not scale well, requires central administration and presents a single point of failure.

By *peer-to-peer* (P2P) networking one makes an attempt to solve the problems of the client/server model by constructing a system in which all users act as both clients and servers. Participants in a peer-to-peer network share their own resources, including information, bandwidth, processing power etc., in a distributed manner. The benefits of this approach are the efficient use of existing resources, scalability, and reliability.

P2P networks have evolved with regard to the level of decentralization. The first well-known P2P protocol was Napster, in which the actual file transfers were made directly between peers but a centralized server was used in searching for the files. This type of the P2P network is referred to as a hybrid network. Gnutella is an example of a pure, fully decentralized P2P system; a peer asks a neighbor for the desired file. If the neighbor does not have the file, it sends the request forward until the file is found. Finally, the Kazaa network combines the features of Napster and Gnutella by having so-called super-peers acting as local servers for file queries.

Peers communicating with each other form an overlay network. In the Gnutella network the overlay is formed by joining the nodes randomly. This kind of random overlay, referred to as an unstructured P2P network, does

not use any routing information, and can lead to poor performance because of the huge control traffic. In structured P2P architectures, on the other hand, the overlay topology is formed in such a way that queries are easier and more effective. The structure can vary from loosely to highly controlled systems [LCC+02]. An example of a loose system is Freenet, in which the overlay topology is formed on the basis of empirical information obtained from peers. Meanwhile, Distributed Hash Tables (DHTs) is a common name for the highly structured information management systems in which each peer is responsible for a set of *keys*, which are mapped to some *value*, such as an address or a file. An example of a DHTs is Chord [SMLNDK03], in which the keys are structured as a ring and searches are made on the basis of the location in the ring.

The basic idea of P2P networks is that the peers do not only use resources but also provide them. However, it is hard to force peers to cooperate without central control. The BitTorrent protocol, a description of which can be found in [Coh03], was developed to overcome these difficulties.

In BitTorrent relatively large files, such as movies, are shared quickly in a decentralized manner. First, the file to be distributed is divided into parts, the sizes of which are typically 256 KB. When a peer wants to download a certain file, it connects to a tracker, which is a centralized server that delivers information about the peers. The tracker returns a list of potential peers to the downloader. Then the downloader contacts other peers and asks which pieces of the file they have. If the peers have the desired pieces, the downloader then requests them. After the peer has downloaded the first piece of the file, it can serve other peers by uploading the piece as a leecher. When the peer has downloaded all the parts of the file, it may stay in the system as a *seed*.

In BitTorrent willingness to upload is ensured by a *tit-for-tat* mechanism. The peers can upload to only four peers at a time. When more requests arrive, the peers for which the chunks are uploaded are selected from those peers that provide the highest download rate for the given peer. The procedure is called as unchoking. To ensure that the peers that do not have any chunk yet can also download the first one, an optimistic unchoking procedure is used: every 30 seconds the peer drops the worst downloader in terms of provided download rate and selects a new, random one.

It has been widely reported that P2P-related traffic forms a significant part of the total traffic in the Internet and that the share is even increasing [KBB+04], [Cac]. The reported shares vary from 50% to 80% depending on the direction of the link. However, there are two major difficulties in measuring P2P traffic and giving exact shares of the link traffic it accounts for. First, many P2P protocols use variable ports or ports designated for some traditional applications and thus identification of P2P traffic is difficult. Second, the evolution of P2P protocols and applications is very rapid and, even more, the popularity of a specific protocol depends heavily on the country. For example, in 2005 BitTorrent was the most used P2P application in Scandinavia, whereas eDonkey was popular in southern Europe [Cac].

1.6 Outline of the Thesis

In this thesis we study traffic engineering in the Internet in three different parts. We start by characterizing traffic measurements gathered in a link of the Funet network. The moving IID Gaussian traffic model is tested with both aggregate link traffic and OD pair representatives. The results of this traffic characterization part of the thesis can be used as input for different functions of traffic engineering. Next, load balancing mechanisms in various types of networks are studied. We propose an adaptive load balancing algorithm both for MPLS and OSPF networks and evaluate the performance of the algorithm. The load balancing problem with service differentiation by WFQ scheduling and load balancing in wireless mesh networks is also studied. Finally, we study the performance of file sharing in peer-to-peer networks, focusing on the population dynamics of downloaders and seeds, the mean lifetime of a file-sharing process, and resource usage of the underlying network. First we study a one-chunk Markov model with absorption and then extend the model to cover two chunks. Simulations are also used to verify the analytical results.

The three areas of traffic engineering studied in this thesis are strongly connected to each other. A major part of the traffic in the current backbone networks is produced by p2p file sharing applications. The characteristics of this traffic can be found by measurement studies. Finally, detected congestion of the links can be alleviated by load balancing.

The thesis is organized as follows: in Chapter 2, we first commence our study by presenting the traffic matrix estimation problem, which is strongly related to traffic engineering and, especially, to load balancing. Then we introduce measurements made in a Finnish university network Funet connecting the universities and many research institutions. Our main purpose is to evaluate the correctness of the moving IID Gaussian model against the link data of the Funet network with different levels of temporal and spatial traffic aggregation. In addition, we are interested in the diurnal variation of traffic and the existence of a specific mean-variance relationship. The results of the measurement study can be used in the evaluation and validation of the traffic engineering mechanisms.

In Chapter 3, we study load balancing, which is probably the main application of traffic engineering. First the familiar static load balancing problem for communication networks is formulated and then a novel algorithm, which is both adaptive and distributed, to solve this problem is proposed. The algorithm is applied to both MPLS and OSPF networks and the obtained results are compared. We continue studying load balancing by formulating an optimization problem that differentiates traffic classes by means of both routing and scheduling. Finally, the formulation of the load balancing problem is modified for Wireless Multihop Networks, in which capacities of the links are not fixed but can be changed, within certain limits, by lower-layer link scheduling.

In Chapter 4, the performance of P2P networks is studied by evaluating BitTorrent-like file-sharing systems. We are especially interested in the lifetime of the file sharing process and the mean download time. To evaluate those metrics, a detailed Markov chain model for the distribution of a single

chunk is proposed for both deterministic and time-dependent peer arrival rates. After that a Markov model for two chunks is proposed, allowing us to compare different chunk selection policies. The analytical results are complemented with simulations. Finally, we propose a spatio-temporal model for a P2P file-sharing system and derive analytical bounds for the resource usage of the underlying network with two different peer selection policies.

2 CHARACTERIZING INTERNET TRAFFIC FOR TRAFFIC ENGINEERING PURPOSES

2.1 Introduction

An essential part of planning and managing operational networks is the measurement process. By measurements the operators obtain important information about network conditions such as congestion and link failures, enabling them to execute corrective actions.

Measurements can be classified in many ways. *Location* of measurement indicates which part or element of the network is being measured and exactly what quantity is being measured. *Time scale* of measurement refers to the length of the measurement period. Moreover, measurement can be continuous or sample-based. The measurements can also be categorized into passive and active: in active measurements a probe packet is sent to the network and the corresponding response is measured, whereas passive measurements do not increase the actual load of the network.

Data collected by measurements are used for different purposes, such as traffic characterization, network monitoring, and traffic control [LTB03]. Traffic characterization refers to the identification of traffic patterns, distributions and trends. Network monitoring determines the state of the network, such as a fault in some network element, and observes the quality of network services. The purpose of traffic control is to react to changes in the network state on a short time scale by mechanisms such as rerouting, resource reservation and admission control.

Aggregation of the measured data is a fundamental part of traffic analysis. Aggregation occurs on two levels: first, measurements have to be made with a certain granularity and second, results are also presented with some defined granularity [Cla94]. In addition, the measured data can be aggregated in time and space. Diurnal variations in the Internet traffic are usually studied at the coarse level of temporal aggregation, with a sample interval of some minutes, whereas the packet-level dynamics ave to be studied with a very fine time granularity. Studies related to the traffic flowing between two hosts are examples of fine-level spatial aggregation, whereas ISP-level studies are examples of coarse aggregation in space. Aggregation in both the measurement and analysis phases is essential to keep the data volumes manageable.

In traffic engineering there is special interests in Origin Destination-based (OD) traffic. The OD flow is a collection of traffic that enters the network from some common ingress point and exits from a common egress point [LPC+04]. For example, in load balancing the network-wide view of traffic is desirable for making routing more efficient. The identification of large single OD flows can also be utilized in adaptive load balancing approaches. On the other hand, the aim of traffic matrix estimation is to provide end-to-end traffic demands from measured link loads. The development of estimation techniques is not possible without knowledge of the statistical properties of the traffic between OD pairs.

In this chapter we characterize the Internet traffic measured in the Funet network for traffic engineering purposes. First the aggregate link traffic is considered and then link traffic is broken down into many OD pair components. We find that OD pairs differ significantly from each other and four different typical OD pair representatives are identified called "Normal", "Bursty", "Uniform", and "Periodic".

We are specially interested in how the so-called moving IID Gaussian model fits to our data. The model includes assumptions about the distribution of the traffic as well as the independence of the consecutive samples of the trace. Our conclusion is that the Gaussian approximation is valid for aggregate link data and some representatives of OD pairs, whereas the IID assumption is questionable for most of the traffic samples.

2.2 Gathering information from the Internet

The most common way to make passive measurements in IP networks is use of Simple Network Management Protocol (SNMP) standardized by IETF in [CFSD90]. SNMP defines how communication between different elements of the network is performed. Typically, the SNMP measurement provides the average traffic load of a link calculated over a 5 minutes period.

Packet monitoring refers to an approach in which a monitor records a copy of the packet traversing through a given link. As an example, `tcpdump` software records IP packets for further analysis. However, recording packet information in the high-speed links is difficult without a dedicated hardware, and also very expensive due to the huge amount of data. Three methods to reduce the huge traffic volumes produced by measurements are aggregation, filtering and sampling [Duf04].

Sampling methods can be categorized to systematic sampling, simple random and stratified sampling [Duf04]. In the most simple approach, systematic sampling, every nth object is selected, where n is the length of the sample period. Whereas systematic sampling is very vulnerable to a bias, random additive approach improves sampling by using sample intervals with random lengths. Finally, a uniform stratified random sampling divides the measured objects into strata and the samples are taken randomly from each stratum.

Detailed information produced by packet monitoring can be very useful in the network management. Fraleigh et al. [FML+03] try to get better understanding of network dynamics by developing further an IP Monitoring (IPMON) system, which measures packet-level statistics in the Sprint IP backbone network. By the IPMON system it is possible to collect packet-level traces over a period of several days on the backbone links, mark each packet with a submicrosecond timestamp, and synchronize timestamps.

Butenweg describes a decentralized measurement system in MPLS networks for traffic engineering purposes in [But03]. In the system each Label Switched Router (LSR) monitors the total load of its outgoing links over a certain time period and calculates the average load of each outgoing link. In addition, the traffic load of each LSP passing through the router is monitored and the average load is calculated.

Active measurements are performed by sending probe packets and an-

alyzing the response of the network. By this approach one tries to figure out the end-to-end characteristics between two network elements. Typically the network operator or the end user is interested in the availability of some service, delay, or packet loss. Internet Control Message Protocol (ICMP) is available at each router and host and is thus a common protocol used to manage active monitoring and specially to get feedback about network problems such as link failures. However, sending traffic to the network has an influence on the performance metrics and this has to be taken into account in the analysis of the measurements. Nevertheless, as compared to passive measurement methods like packet monitoring, active methods require less computational power or hardware updates.

2.3 Origin-destination pair traffic in the Internet

In managing operational networks it is important to have a network-wide view of the traffic. Indeed, many traffic engineering mechanisms assume that the traffic matrix is available. However, it is well-known that the information of point-to-point traffic demands is not directly available in IP networks. Measuring the link loads of the network is quite straightforward, but the obtained loads do not only indicate the traffic demands but also the decisions made by the current routing protocol. To find out the offered load between each origin and destination node, some estimation techniques have to be used instead.

There are four main approaches to derive the traffic matrix [FRT02]. The first one is a direct use of SNMP data. One can define the traffic volumes on Label Switched Paths (LSP) in MPLS network by MPLS MIBs, for example. The second approach is to combine packet-level and flow-level measurements with the routing table information. The third approach tries to estimate the traffic matrix from the measured link loads, whereas the fourth approach is based on directly sampling observed traffic flowing through the network. The second and third approaches are explained in more detail in the paragraphs below.

One of the earliest studies in deriving traffic demands in an IP backbone was presented by Feldmann et al. in [FGL+01]. Instead of estimating point-to-point traffic demands, according to the paper, it is more preferable to model point-to-multipoint traffic volumes, which fits better to the nature of IP routing. An ISP cannot control the ingress link of traffic but the choice of the egress link depends on both intradomain and interdomain routing parameters and thus point-to-point traffic demands depend on the routing decisions. The traffic demands in [FGL+01] are computed from the measured flow-level statistics. The starting and finishing time, the total number of bytes and the input link and destination address of a flow are collected at the ingress routers. To derive the point-to-multipoint traffic volumes of those measurements, the destination prefix of each flow is associated with a set of egress links.

Three different approaches to estimate the traffic matrix from the link loads are introduced and compared in [MTS+02]. The problem formulation is as follows: Let Y be the vector of link measurements and A the routing matrix where element $A_{lk} = 1$, if OD pair k uses link l, and 0 oth-

erwise. Let X be the traffic demand vector, in which element X_k denotes the traffic demand of OD pair k. The following relation connects the link measurements to the traffic demands:

$$Y = AX.$$

The traffic demands cannot be solved directly from the equation above since the number of OD pairs is usually much larger than the number of links. The three techniques for solving this under-defined linear problem listed in [MTS+02] are: 1) use of linear programming to find maximal demands satisfying the link constraints, 2) Bayesian Inference technique and 3) approach based on Expected Maximization (EM) algorithm to calculate the maximum likelihood estimates.

2.4 Related research

Characterization of the Internet traffic has gained a lot of interest during last ten years. A common belief is that the Internet traffic differs significantly from the traffic of traditional telephone networks and cannot thus be modelled by the familiar Poisson model. Instead, traffic has found to be self-similar, meaning that aggregation of traffic does not necessarily smooth the existing variability. In addition, many measurements have shown traffic to be heavy-tailed. Most of the earlier studies concentrate on the aggregation in time, whereas spatial aggregation has not attained as much attention.

Characterization of aggregate link traffic

Leland et al. presented one of the earliest and best-known Internet measurement studies in [LW91] and [LTWW94]. In the papers, traffic of Ethernet local area network in Bellcore is characterized and found to be self-similar; the traffic trace seems to have similar bursts on many time scales. In later works, characterization of Internet traffic has been extended to the wide area networks. Claffy characterizes the traffic of NSFNET backbone by a flow-based approach in [Cla94]. Paxson and Floyd continue the work of Leland et al. in [PF95] and [Pax97], and find that in wide area networks the Poisson model is valid for the arrival process of new user sessions but not for the packet arrival process due to self-similarity of traffic. The reason behind the self-similarity has been found to be in the heavy tail distribution of the file size [PKC96].

After the observation of inherent self-similarity in Internet traffic many traffic models have been proposed to characterize this phenomenon. One of the earliest models is the fractional Brownian motion model proposed by Norros for communication networks in [Nor94], [Nor95].

In addition to self-similarity and heavy-tailed file size distribution, Internet traffic has been reported to be non-stationary ([MD00], [CCLS01], [Uhl04]). Karagiannis et al. point out in [KMFB04] that due to the huge growth of Internet traffic in recent years, the validity of ten year old assumptions has to be investigated again. The result from a measurement study made in 2002 is that on the short time scale (microseconds) the packet inter-arrival time distribution can be approximated by an exponential distri-

bution. On a longer time scale traffic seems to be nonstationary, although it is smoother than in the three year older measurements.

Traffic engineering deals with network optimization tasks mainly for the aggregated traffic, not for individual flows. The Central Limit Theorem (CLT) states that when the traffic of large number of independent users is aggregated, the resulting traffic process is approximately normal. Gaussian assumptions are tested against Internet data in many studies such as [KN02] and [vdMMP06]. Kilpi and Norros study in [KN02] how much traffic has to be aggregated in vertical and horizontal direction in order for the resulting aggregate to be Gaussian. Traffic traces in their study are from an access network, resulting in rather low traffic rates. The conclusion is that for downstream traffic Gaussian model fits well on many time scales, whereas for upstream direction and time scales over 0.2 seconds log-normal distribution is better. Van de Meent et al. [vdMMP06] continue the work of [KN02] and claim that if traffic is Gaussian on one time scale, it is Gaussian on other time scales as well.

Characterization of OD pair traffic

Characterization of origin-destination based traffic has received less attention in recent measurement studies. Although estimation of the traffic matrix in the Internet is a widely studied research subject, full understanding of OD pair traffic is lacking.

In Section 2.3 we reviewed some new measurement methodologies developed by Feldmann et al. [FGL+01]. In addition to this, they also characterize point-to-multipoint traffic in the same paper. The result is that a few demands account for 80% of total traffic and the traffic volumes follow Zipf's law. Daily profiles of the greatest demands also vary much.

Bhattacharyya et al. characterize Point of Presence-level (POP) and access-link level traffic dynamics in [BDJT01]. Also they find that there are huge differences in the traffic volumes of the demands involving traffic from a single ingress node to many egress nodes. The larger the traffic volume of an egress node, the larger also the variability of traffic during the day.

Lakhina et al. [LPC+04] analyze traffic of two backbone networks, one from Europe and one from the US. Using Principal Component Analysis (PCA) they demonstrate that OD flows can be approximated by a linear combination of a small number of so-called eigenflows. In addition they observe that these eigenflows fall into three categories: *deterministic* eigenflows exhibiting strong diurnal periodicity, *spike* eigenflows with clear outliers, and *noise* eigenflows with a nearly Gaussian marginal distribution.

Cao et al. propose a *moving IID Gaussian model* for OD traffic flows in [CDWY00]. In the model the flow x_{nk} of OD pair k at time moment n is a realization of corresponding stochastic process $(X_{nk}; n = 1, \ldots, N)$, which consists of a deterministic term $E[X_{nk}]$ capturing the possible cyclostationary diurnal pattern and a randomly fluctuating term $D[X_{nk}]Z_{nk}$:

$$X_{nk} = E[X_{nk}] + D[X_{nk}]Z_{nk}, \tag{2.1}$$

where the standardized residuals,

$$Z_{nk} = \frac{X_{nk} - E[X_{nk}]}{D[X_{nk}]}, \tag{2.2}$$

are assumed to be independent and identically distributed according to a normal distribution with zero mean and unit variance. The validity of the model is tested in a local network at Lucent with a measurement period of 300 s. They conclude that the Gaussian model is valid, even though we find their data to be very bursty and far from Gaussian.

Along with the traffic matrix estimation studies, a question whether there is a structural relationship between the mean and the variance of an OD flow, has emerged. If the flows are modelled by the Poisson model (as assumed in [Var96], for example) the variance equals the mean:

$$D^2[X_{nk}] = E[X_{nk}].$$

Also Cao et al. assume in their moving IID Gaussian model a specific *mean-variance relationship*,

$$D^2[X_{nk}] = \phi E[X_{nk}]^c. \tag{2.3}$$

The exponent c is scale-invariant, while the factor ϕ depends on the data unit used (for a more detailed discussion about scale-dependence and scale-invariance, see Publication 1). Cao et al. reports that a quadratic power law ($c = 2$) is reasonable for their data. Medina et al. [MTS+02] find that the exponent c varies significantly from link to link ($c \in [0.5, 4]$) and Gunnar et. al [GJT04] report that values $c = 1.5$ to $c = 1.6$ fit to their data. As a conclusion, there is no consensus about the existence of the mean-variance relationship or about the value of parameters c.

In addition to aforementioned studies, there exist also some papers that are more loosely related to our work. First, the spatial distribution of traffic in a French academic network is studied in [BGG04]. The result of the paper is that the most of the traffic flows between a few active nodes in the network. The purpose of the paper [ZZB05] is to profile the Internet backbone traffic by comprehensive communication patterns between end-hosts and ports. Finally, in [FVFB05] geographic location of IP prefixes is studied and the finding is that the networks having the same network prefix (24 bits) can actually be located very far from each other. This naturally affects the efficiency of load balancing and other traffic engineering mechanisms.

2.5 Contribution

As a contribution on the measurement part in this thesis, we characterize a backbone network traffic for traffic engineering purposes. Common assumptions, such as Gaussianity and independence of the traffic samples, are studied with different levels of spatial and temporal aggregation of traffic, starting from the total link traffic and ending to four representatives of typical OD pair traffic.

Publication 1 introduces for the first time the measurements made in the Finnish University network (Funet). In the paper, the moving Gaussian IID model as well as the mean-variance relationship are tested against

the Funet data at different levels of temporal aggregation, from one second to 5 minutes. The results show that the model is better applicable to current backbone networks than to the local area networks studied earlier. However, due to autocorrelation, IID assumption is not valid.

Publication 2 continues the investigation of the moving Gaussian IID model by splitting the link traffic into OD pairs based on the source and destination network prefix of the IP packets. With prefix length $l = 22$ we identify four different types of OD pairs, namely "Normal", "Bursty", "Uniform" and "Periodic". Stochastic properties of these types are studied and it is found that only the "Normal" OD pair can be approximated by Gaussian distribution. IID assumption is instead best suited for "Uniform" OD pair. OD pairs even with different origin and destination networks are found to be correlated, which probably induces errors to many traffic matrix estimation techniques.

Finally, in Publication 3 the OD pair traffic is studied further. We investigate how spatial aggregation of the data based on the network prefix affects the characteristics of OD pair traffic. Our finding is that power-law assumption is valid for fine aggregation of traffic whereas the Gaussian approximation is better for coarser granularity.

In next sections we go through the details of the measurement study done in the Funet and present the main results of the papers. In this chapter we concentrate on the validity of the moving Gaussian IID model. Numerical results related to the mean-variance relationship can be found in the above publications.

2.6 Characterizing traffic of the Funet network

In this section we present the measurement methodology and the obtained data series. The traces of the total link traffic are illustrated and the volumes of the OD pairs with different spatial aggregation levels studied. Finally, four representative OD pair types at a specific spatial aggregation level are identified.

Measurements

Traffic traces used in this thesis were captured by Endace DAG 4.23 cards from 2.5 Gbit/s STM-16 link connecting nodes csc0-rtr and helsinki0-rtr in the Finnish university network (Funet) connecting all the Finnish universities and other research institutes [1]. The link is two-directional: We denote by d_0 the direction from csc0-rtr to helsinki-rtr and by d_1 the opposite direction. In this thesis we consider the traffic traces from July, August and November 2004.

The measurement methodology was the following: The IP addresses on all packet headers in the link were anonymized preserving headers of the packets. The headers were then stored to disk using flow-based compression [Peu01]. Captured data was transferred once an hour to an analysis machine, where statistics were calculated. Because the volume of the data produced by the measurement process was massive (about 10 Mbps on average), only part of the traces were archived for later analysis.

[1] Details about Funet can be found from http://www.csc.fi/suomi/funet/verkko.html.en

TCP accounted for more than 98 % of bytes transferred in the link. During daytime 10-20 % of TCP traffic was HTTP (TCP port 80). There existed also a considerable amount of peer-to-peer traffic. More details of the applications loading the measured link can be found from a port based analysis of the traffic in [IL06].

The captured traffic of the link was divided into origin-destination pairs by identifying the origin and destination networks of packets by the left-most bits in the IP address. Let l denote the number of bits in this network prefix, also called network mask. Different levels of aggregation are obtained by changing the prefix length l. The maximum length of the network prefix is 24 bits. With this resolution, there are 2^{24}, i.e., over sixteen million possible origin networks. On the other hand, with the prefix length $l = 1$ there are only two networks and thus four possible OD pairs. For some specific prefix lengths, such as $l = 22$, 100 greatest origin and destination networks were selected and the traffic matrix for these networks was constructed resulting in 10000 OD pairs. For other prefix lengths, we considered only 100 most active OD pairs in terms of the traffic volume.

The original traffic and its derivatives

Let us consider the measured traffic of the link. The traffic is divided into origin-destination pairs as explained in the previous sub-section. For each prefix length l and direction d_0/d_1, the original measurement data is denoted by $\mathbf{x} = (x_{t,k}; t = 1, 2, \ldots, T, k = 1, 2, ..., K)$, where $x_{t,k}$ refers to the measured link bit count of OD pair k at time t seconds (from the beginning of the measurement period). For each OD pair we define the volume X_k which tells how much traffic is sent over the measurement period on average:

$$X_k = \frac{1}{T} \sum_{t=1}^{T} x_{t,k}.$$

The original link traffic is aggregated in time with different time scales of Δ seconds, where $\Delta = 1, ..., 300$. For each investigated time scale, we create the corresponding time series of link bit counts $x_k^\Delta = (x_{n,k}^\Delta; n = 1, 2, \ldots, T/\Delta)$ by defining

$$x_{n,k}^\Delta = \frac{1}{\Delta} \sum_{t=(n-1)\Delta+1}^{n\Delta} x_{t,k}.$$

Similarly as in [CDWY00], we separate different components from the link bit counts $x_{n,k}^\Delta$,

$$x_{n,k}^\Delta = m_{n,k}^\Delta + s_{n,k}^\Delta z_{n,k}^\Delta,$$

where $m_{n,k}^\Delta$ refers to the moving sample-average, $s_{n,k}^\Delta$ to the moving sample-standard-deviation, and $z_{n,k}^\Delta$ to the sample-standardized residual. Based on the visual plot of the trace, the averaging period was chosen to be (about)

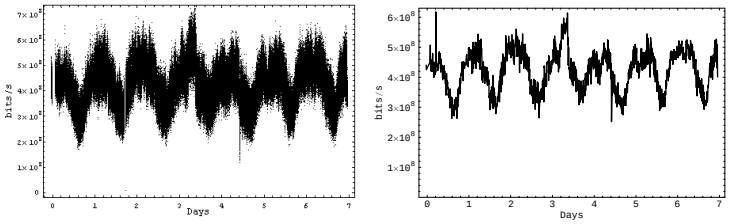

Figure 2.1: Trace of the aggregate link data with $\Delta = 1$ (left) and $\Delta = 300$ (right) over one week.

1 hour. Thus,

$$m_{n,k}^{\Delta} = \frac{1}{3600/\Delta + 1} \sum_{j=n-1800/\Delta}^{n+1800/\Delta} x_{j,k}^{\Delta}$$

and

$$s_{n,k}^{\Delta} = \sqrt{\frac{1}{3600/\Delta + 1} \sum_{j=n-1800/\Delta}^{n+1800/\Delta} (x_{j,k}^{\Delta} - m_{j,k}^{\Delta})^2}.$$

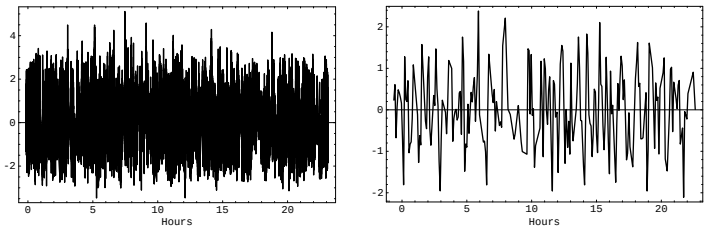

Figure 2.2: The sample-standardized residual z_n^{Δ} of the aggregate link data with $\Delta = 1$ (left) and $\Delta = 300$ (right).

The original trace of the total link traffic over a 7-day period is shown for 1-second and 5-minute time scales on the left and right hand sides of Figure 2.1, respectively. In this figure, as well as elsewhere in this thesis we consider only the direction d_0 of the link. The diurnal variation of the traffic at this high level of spatial aggregation is clearly visible. The busiest hour of the day is usually before the middle of the day, from 11 a.m. to 12 a.m. In Figure 2.2 the sample-standardized residuals of the trace for 1-second and 5-minute time scales are plotted for one selected day. On the 1-second time scale the residuals seem to be random noise, which is not true for the longer, 5-minute time scale.

Next we consider the volumes of the OD pairs with different level of spatial aggregation. One question is whether there is a power law behavior (also known as Zipf's law) observable in the volumes of the OD flows. If the law holds, the volumes of the ranked OD pairs should decrease linearly

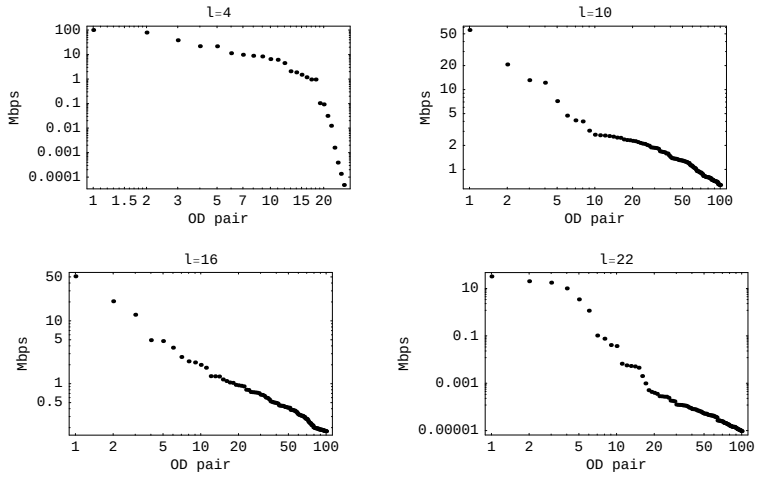

Figure 2.3: Traffic volumes of OD pairs for different prefix lengths l.

on the log-log scale. The OD pairs are sorted from greatest to smallest and their volumes are plotted on the log-log scale in Figure 2.3, when the prefix length varies from $l = 4$ to $l = 22$. For every level of spatial aggregation, i.e., prefix length l, there are approximately 15 very great OD pairs and beyond those the volumes decrease steeply. We note that the decrease is linear and the power-law assumption is probably valid only for finer granularity of aggregation, such as $l \geq 10$, where the traffic volumes are considerably small.

OD pair grouping

With a glance on the OD traffic traces with a specific prefix length, it is obvious that the OD pairs behave differently and it is hard to model them all with some common distribution. For that reason, in this subsection we try to find some characteristics that are common to some set of the OD pairs and group them according to those characteristics. We focus on the traffic that is divided into OD pairs according to prefix length $l = 22$. By this choice each origin subnetwork could have 10 bits for host part or about 1000 IP addresses. Our main finding are summarized below.

First, some OD pairs, as the largest OD pair in terms of the traffic volume (depicted in the upmost row of Figure 2.4), seem to approximately follow the Gaussian distribution. The original trace of this OD pair has a long-term variation, which, however, does not follow the ordinary diurnal pattern since traffic of the OD pair is at its highest at night and lowest in the day of Finnish time. This greatest OD pair constitutes 10 % of the total link traffic.

Secondly, there are some very bursty OD pairs (see the second row of Figure 2.4). During some time periods the traffic load is many times higher and variable than during other periods. Note that there is a significant number of idle seconds in the original trace. For this reason also the noise

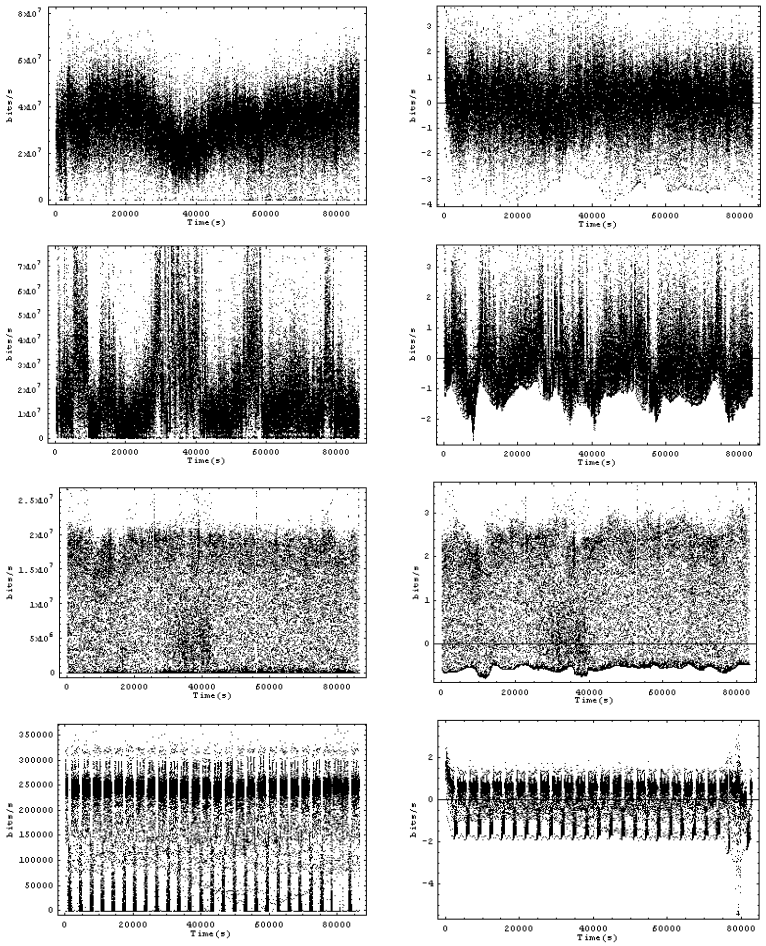

Figure 2.4: Original traffic trace (bits/s) $x_{n,k}^{\Delta}$ in the left side and residual component $z_{n,k}^{\Delta}$ in the right side of figure. From top to bottom: *Normal*, *Bursty*, *Uniform* and *Periodic* OD pair. Time scale $\Delta = 1$.

of the residual component is not symmetrically distributed around zero, but has a clear lower bound.

Thirdly, some OD pairs seem to follow approximately the uniform distribution. This type of OD pair is depicted in the third row of Figure 2.4. The traffic trace is concentrated on zero and peak rates (approx. 20 Mbits/s), but there are also a lot of samples between those rates.

Finally, some OD pairs are clearly periodic, as the last OD pair in Figure 2.4. The period of the representative OD pair is roughly one hour long. Note that the maximum rate of this OD pair is much smaller than the corresponding rates of the three earlier examples.

At this level of spatial aggregation none of the example OD pairs, named "Normal", "Bursty", "Uniform", and "Periodic", seem to follow the diurnal pattern of the aggregate link traffic. This indicates that periodic variation

in the aggregate link traffic is better explained by the changes in the (large) number of small on-off traffic flows than by the changes in the volumes of the large OD pairs. Small OD flows have idle periods more probably during night time than day time and thus the total traffic volume of the link is smaller at night.

2.7 Diurnal variation of the Funet data

In the previous section we observed that at the aggregation level of $l = 22$ none of the OD pairs seemed to follow the diurnal variation of the total link traffic, in which the traffic peak is in the midday. Our explanation was that the strong diurnal variation in the link traffic is more explained by the variation in the number of active on-off OD pairs than diurnal pattern within these OD pairs. However, we would expect that when the spatial aggregation of the traffic is increased by decreasing l, at some point the standard diurnal pattern should become visible in the traces of the OD pairs. As an example, we plot the moving sample averages of the four largest OD pairs with aggregation levels $l = 4$ and $l = 8$ in Figure 2.5. At the coarse level of aggregation we can see different types of diurnal patterns. Pairs 3 and 4 have a diurnal variation close to the variation of the total link traffic, while pairs 1 and 2 are not so close. At the resolution $l = 8$ only the third OD pair follows somehow the diurnal pattern of the link traffic.

To understand better how the diurnal traffic variation behaves as the spatial aggregation level changes, we calculate the correlation between two time series: the moving sample average of the total link traffic, and the moving sample average of OD pair k at aggregation level l. If the correlation between these two time series is high, also the diurnal traffic patterns are similar. The correlation coefficient between any two time series $x = (x_i, i = 1, ..., n)$ and $y = (y_i, i = 1, ..., n)$ is defined as

$$r(x, y) = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}}. \tag{2.4}$$

On the left hand side of Figure 2.6, we plot the correlation coefficients for all OD pairs with all aggregation levels l and directions d_0 and d_1 as a function of the volume of the OD pair. For small OD pairs there exists both positive and negative correlations but for large OD pairs the correlations are positive, as we would be expected. However, dependence between the correlation and the volume of the OD pair is not strong. On the right hand side of the same figure, the mean of the correlation coefficients for the OD pairs with given prefix length l are plotted. We can see that the mean correlation decreases as a function of l, but the decrease is not necessarily monotonic.

As a conclusion, we can argue that as the aggregation level of the traffic is increased, also the diurnal pattern of the OD pairs is closer to the that of the total link traffic. However, there is no clear bound in OD pair volume or in spatial aggregation for this effect.

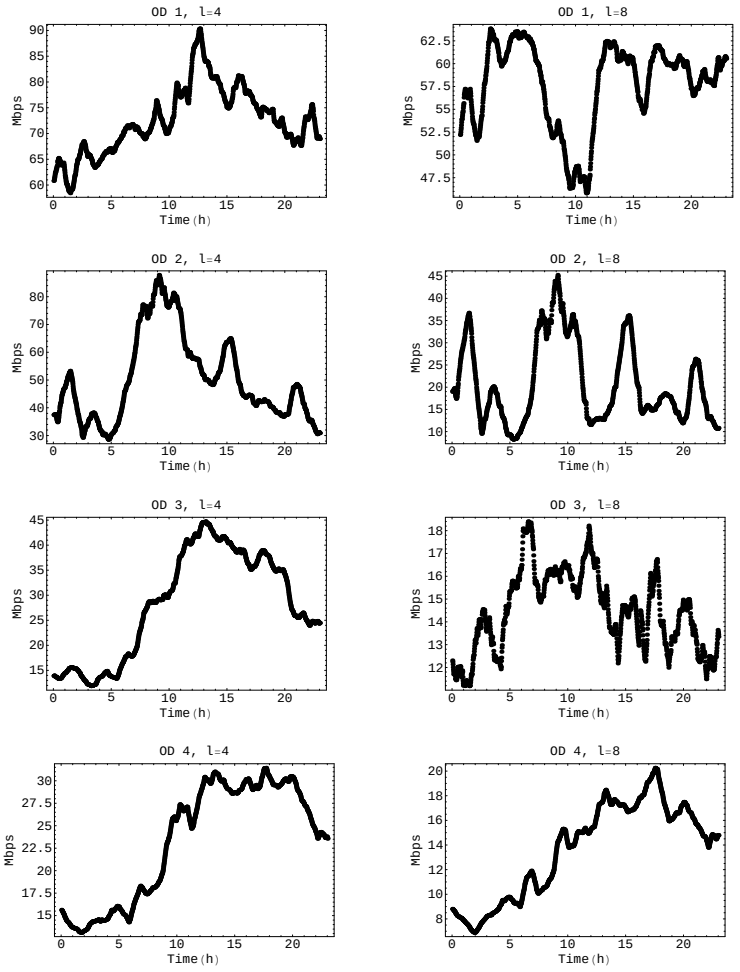

Figure 2.5: The moving sample average for the 4 greatest OD pairs. Prefix length $l = 4$ (upper graphs) and $l = 8$ (lower graphs).

2.8 Validity of the moving Gaussian IID model

In this section we study the validity of the moving Gaussian IID model introduced in Section 2.4. The model assumes that the traffic of the OD pairs follows the Gaussian distribution and both consecutive bit counts of a given OD pair and bit counts over two different OD pairs are independent (IID assumption).

One way to evaluate the appropriateness of the Gaussian approximation for some data is to use the normal quantile (N-Q) plot. The original sample vector ordered from the smallest to the largest, denoted by x, is plotted

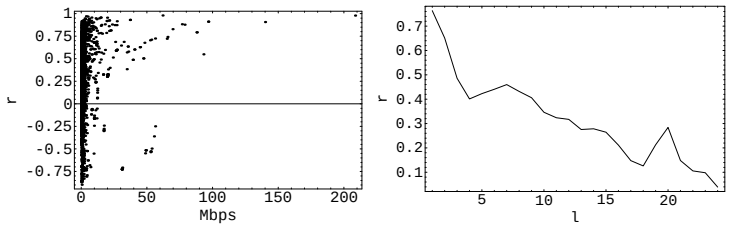

Figure 2.6: Left: The correlation between the moving sample averages of the total link traffic and an OD pair as a function of the traffic volume. Right: Average correlation as a function of prefix length l.

against quantile vector a defined as:

$$a_i = \Phi^{-1}(\frac{i}{n+1}) \qquad i = 1, \ldots, n,$$

where Φ is the cumulative distribution function of the normal distribution with zero mean and unit variance. If the considered data follow the normal distribution, the resulting plot $\{x_i, a_i\}, i = 1, ..., n$, is linear. Goodness of the Gaussian approximation can be evaluated by the square of the correlation coefficient $r(x, a)$, where r is defined similarly as in equation (2.4).

If the traffic samples of an OD pair are independent, there should not be any significant autocorrelation in the residual components of the flow. The autocorrelation function for residual $z^{\Delta}_{n,k}$ is defined as:

$$r_l(k) = \frac{\sum_{i=1}^{T/\Delta - l}(z^{\Delta}_{i,k} - \bar{z}^{\Delta}_{k})(z^{\Delta}_{i+l,k} - \bar{z}^{\Delta}_{k})}{\sum_{i=1}^{T/\Delta}(z^{\Delta}_{i,k} - \bar{z}^{\Delta}_{k})^2},$$

where T/Δ is the size of time series and l is the *lag*.

Finally, in addition to independence of the consecutive samples, also the individual OD pairs should be independent of each other. The dependency between the OD pairs is assessed by calculating the cross-correlation $r(z^{\Delta}_{n,k}, z^{\Delta}_{n,k'})$ between the residuals $z^{\Delta}_{n,k}$ and $z^{\Delta}_{n,k'}$ of different OD pairs k and k', where r is again the standard correlation coefficient defined by (2.4).

Gaussian approximation

First we study Gaussianity of the aggregate link traffic on different time scales. The data include all traffic using the link, meaning maximal spatial aggregation. As an example of two different time scales, on the left hand side of Figure 2.7 the histograms of the standardized residuals of the aggregate link traffic are plotted together with the density function of the normal distribution with the same mean and variance. For the 1-second time scale the data follow Gaussian density function nicely. For the 5-minute time scale there are some outliers, but the fit is still reasonably good. The N-Q plots for the same data sets are shown on the right hand side of Figure 2.7. The r^2-values are 0.999 and 0.996 for the 1-second measurements and the

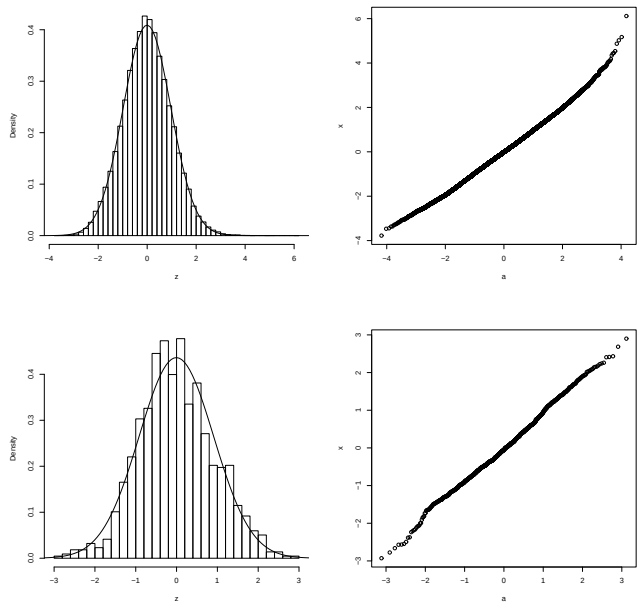

Figure 2.7: Histograms with the normal distribution and N-Q tests of the aggregate link data with $\Delta = 1$ (the upper figures) and $\Delta = 300$ (the lower figures).

5-minute time scales, respectively, confirming Gaussianity observed from the histograms.

Next we study the goodness of the Gaussian approximation separately for the four representative OD pairs presented in the previous section. The marginal distributions and N-Q plots are presented in Figure 2.8. The goodness of the approximations for the representative OD pairs on two time scales, one second and one minute, are collected to Table 2.1. On the 1-second time scale only the "Normal" OD pair seems to follow the Gaussian distribution. "Bursty" OD pair exhibits second best fit. When time scale is increased to one minute, surprisingly "Uniform" OD pair is also quite close to normal distribution.

Third, the effect of the spatial aggregation for the Gaussian approxima-

Table 2.1: Goodness of fit, r^2 for the total link traffic and OD pairs

OD pair	r^2 ($\Delta = 1$)	r^2 ($\Delta = 60$)
Total	0.993	0.993
Normal	0.996	0.996
Bursty	0.936	0.976
Uniform	0.661	0.965
Periodic	0.816	0.698

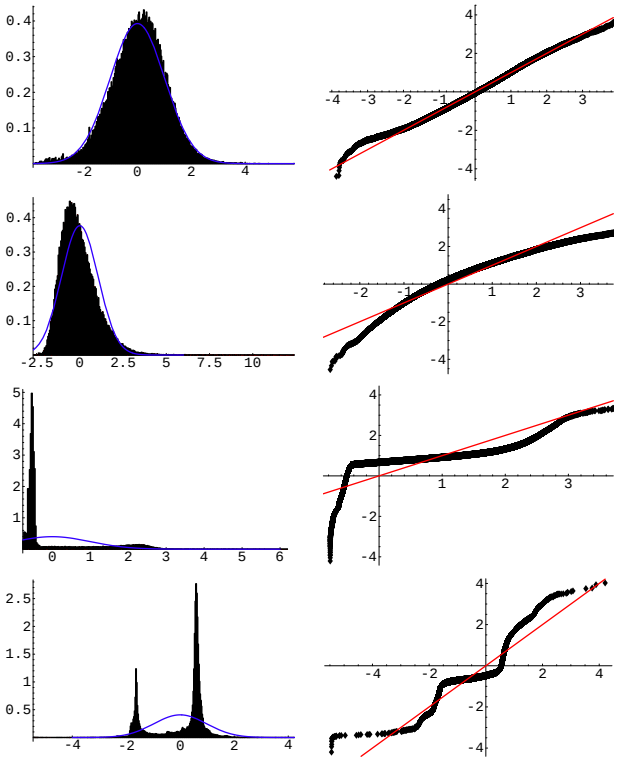

Figure 2.8: Marginal distributions and N-Q plots for residuals with $\Delta = 1$. From top to bottom: *Normal*, *Bursty*, *Uniform* and *Periodic* OD pair.

tion is studied on the 1-minute time scale. To this end, we plot the correlation value r^2 as a function of prefix length l on the left hand side of Figure 2.9. The average value for r^2 is taken over those OD pairs that constitute 80 % of the total traffic. From the figure we can see that r^2 decreases below 0.9 when $l = 8$. Another possibility to examine spatial aggregation is to plot the correlation r^2 as a function of the volume of OD flow as is done on the right hand side of Figure 2.9. For file sizes over 18 Mbps it holds $r^2 > 0.9$.

As a conclusion of this subsection, the Gaussian approximation cannot be rejected as the total link traffic or a "Normal" OD pair is considered. For these examples the goodness of the approximation does not depend on the time scale (r^2 remains constant from the 1-second to 5-minute time scales). Traffic of other representative OD pairs is further from Gaussian on many time scales. An exception is the "Uniform" OD pair, for which aggregation in time clearly improves the Gaussian approximation. When the spatial aggregation is studied in more detail, it is found that the greater the aggregation of the networks or the greater the OD-flow volume, also the better the Gaussian approximation. However, it is not possible to find a specific threshold for Gaussianity.

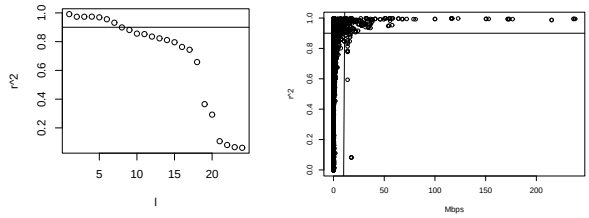

Figure 2.9: The goodness of Gaussian approximation r^2 as a function of the prefix length l (left) and the OD pair volume (right).

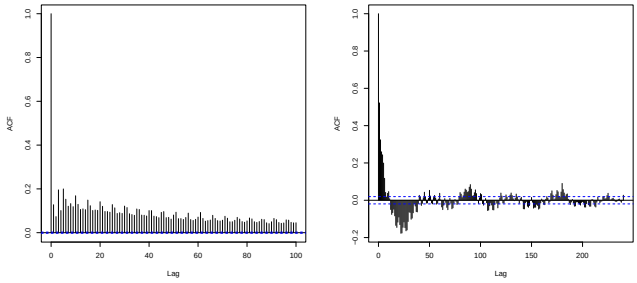

Figure 2.10: Autocorrelations of the aggregate link traffic when $\Delta = 1$ and $\Delta = 60$.

Dependence of the traffic samples

In this subsection we study the dependence of the consecutive measurements at different temporal aggregation levels. First the autocorrelation function (acf) is plotted for standardized residuals z_n^{Δ} of the aggregate link traffic in Figure 2.10. On the 1-second time scale there is a clear positive correlation in the data. We notice also spikes every 5 seconds. On the 1-minute time scale we notice significant positive values in correlation up to a lag of little over five minutes, and then a negative correlation is clearly observable. For lags over 30 minutes there is no autocorrelation anymore.

Next the autocorrelation functions for the standardized residuals of the four representative OD pairs are depicted in Figure 2.11. On the left hand side of the figure the aggregation level is one second ($\Delta = 1$) and on the right hand side it is one minute ($\Delta = 60$). As "Normal" and "Bursty" OD pairs are considered, there are statistically significant autocorrelation values for the lags of five seconds. Recall that the same spikes exist also in the autocorrelation function of the total link traffic in Figure 2.10. A closer inspection of the trace of "Normal" and "Bursty" OD pairs showed that traffic of these OD pairs decreased dramatically every 5 seconds. This behavior is most likely caused by one common origin network that the "Normal" and "Bursty" representative OD pairs share. Since these OD pairs form over

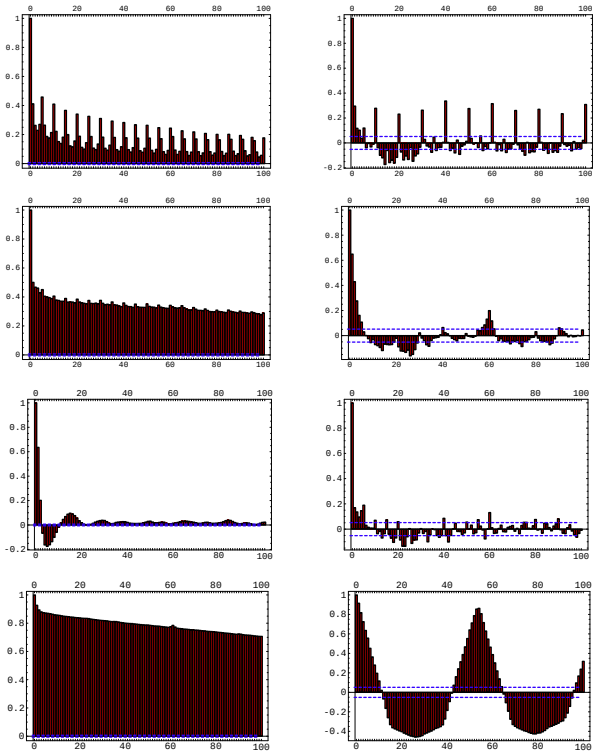

Figure 2.11: Autocorrelation function for residual component. Left side: $\Delta = 1$, right side: $\Delta = 60$. From top to bottom: *Normal*, *Bursty*, *Uniform* and *Periodic* OD pair.

10% of the total link traffic, the spikes in the autocorrelation function of the total link traffic are also explained by the behavior of the same origin network.

For longer lags, the autocorrelation of "Normal" and "Bursty" pairs tends to zero, and is below the confidence interval shown with dashed lines. The exceptions are the spikes in the traces of the "Normal" OD pair at every 10 minutes.

From the "Uniform" OD pair we note that the autocorrelation is almost zero even with very short lags of a few seconds. The central limit theorem requires uncorrelated traffic samples and this explains why only for the "Uniform" OD pair temporal aggregation of the data improved the correctness of Gaussian approximation in the previous subsection.

The last row of Figure 2.11 depicts the autocorrelation function of the "Periodic" OD pair. This is the only OD pair which clearly exhibits long range dependence, in this case because of periodicity. Indeed the periods can be clearly observed from the correlation structure.

As a conclusion of this subsection, the consecutive traffic counts of the aggregate link traffic and OD pair representatives have significant amount of autocorrelation and cannot thus be considered independent. Especially

the "Normal" OD pair has spikes in the autocorrelation function every 5 seconds and 10 minutes. Samples of the "Uniform" OD pair were least correlated.

Dependence between OD pairs

Whereas in the last subsection we studied the dependence of the consecutive samples in the traffic trace of an OD pair, in this subsection dependence between different OD pairs is considered. The assumption of independent OD pairs is essential in many traffic matrix estimation techniques, for example. For studying independence, we select 20 great OD pairs with aggregation level $l = 22$ in terms of the traffic volume and calculate the cross-correlation for the standardized residuals of them. This approach results in 400 correlation values.

The correlation values between pairs of OD pairs are presented graphically in Figure 2.12. In the figure we have excluded the comparison of a given OD pair with itself, which would obviously imply a correlation coefficient of 1.0. The distribution of the various correlation terms is also shown on the right hand side of Figure 2.12. The horizontal lines in the figure depict the 95% confidence interval of the hypothesis that correlation would be zero. Our observation is that a large number of OD pairs are statistically significantly correlated.

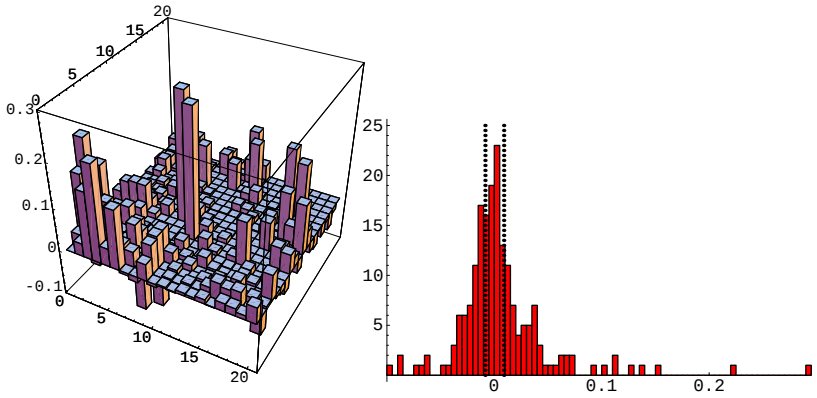

Figure 2.12: Left: Correlation between 20 greatest OD pairs, right: distribution of correlation coefficients, with 95% confidence interval depicted by dotted lines.

For understanding the reason behind the correlated OD pairs we investigate in more detail the 10 pairs of OD pairs that have the greatest correlation value, either positive or negative. Table 2.2 lists those pairs together with the origin and destination networks. From the table we can see that also the pairs that have completely different origin and destination networks can be significantly correlated.

Table 2.2: Origin and destination networks of pairs of OD pairs and cross-correlations between them.

s_1	d_1	s_2	d_2	r
65	42	51	36	0.29
1	5	1	2	0.22
1	4	1	2	0.15
1	21	1	2	0.14
30	5	66	5	0.13
30	46	30	4	0.11
30	5	34	1	0.11
30	5	23	35	0.10
66	5	23	35	0.10
51	36	1	3	-0.10

2.9 Summary and Conclusions

In this section we have studied the characteristics of Internet traffic in a high-speed backbone network connecting Finnish universities. Our work was motivated by the fact that many traffic engineering mechanisms and, especially, traffic matrix estimation approaches, make assumptions about the nature of OD flows. However, these assumptions have not previously been justified in the current backbone networks.

In the study, the traffic of the link was first considered as one aggregate and then classified into OD pairs based on the network prefix of each packet. A small number of large OD flows account for the majority of the link traffic and understanding their characteristics is important. For that reason we identified four different types of OD pair flows and characterized their statistical properties separately. The OD pairs are "Normal", "Bursty", "Uniform", and "Periodic". Our approach in characterizing the different types of OD flows separately cannot be found in the earlier studies and thus gives new insights into the nature of Internet traffic.

We are especially interested in how a moving IID Gaussian model fits to OD pair traffic. Finding this kind of simple traffic models is useful in the evaluation, optimization, and simulation of the contemporary Internet, even if they are only coarse approximations of reality in many cases.

The main result of this chapter is that Gaussian approximation is valid for the aggregate link traffic, as well as for some specific OD pairs, such as a "Normal" OD pair. However, since the consecutive traffic counts are correlated, the IID assumption is not justified. One important point is the correlation between different OD pairs. Even two OD pairs that have completely different origin and destination networks can have a statistically significant correlation. The reason for this can be in some other part of the network, the resources of which these pairs share. The strong correlation of OD pairs has a negative influence on the accuracy of many traffic engineering approaches, such as traffic matrix estimation. In addition, the

correlation of great OD flows causes temporary congestion in the links of the network, which necessitates the use of load balancing mechanisms in the short term.

In this section we concentrated on one single link. Measuring many links concurrently would give some new insights into the problem. However, managing the huge amounts of measurement data, combining the results, and making correct conclusions from them is not a straightforward task. In addition, the boundaries that exist between ISPs make it difficult to create a big picture of OD flows. For these reasons, concentrating on single links is reasonable.

3 LOAD BALANCING MECHANISMS FOR MPLS, OSPF AND WMN NETWORKS

3.1 Introduction

One of the most important applications of traffic engineering is load balancing, the idea of which is to perform congestion management and resource usage optimization by moving traffic from congested links to other areas of the network. As a result of load balancing, when the average link loads are smaller, there are also more resources to handle occasional bursts in the offered traffic of the network.

Successful implementation of load balancing depends on the underlying routing protocol that provides connectivity through the Internet by determining the routes used by traffic flows. An essential property of the routing protocol is the capability to split end-to-end traffic into many parallel paths. Without this property load balancing remains coarse-grained and leads to only a sub-optimal result and, furthermore, is prone to instability.

The answer to the question whether load balancing in IP networks is necessary at all depends largely on the network conditions, that is, the amount of available capacity in comparison with the offered traffic. The task of dimensioning is to ensure that there is enough capacity in the network to serve the demand. According to [FML+03] most of the links are utilized under 50%, but recent measurement studies show that there exist spikes on some links in which the link utilization is over 90% [AA03]. Therefore the use of load balancing, and especially adaptive mechanisms, is desirable.

Adaptive load balancing refers to a system in which network control parameters are automatically adjusted on the basis of changes detected in the state of the network. The changes may be related to traffic patterns, such as point-to-point traffic demands, or to the network topology, such as link or router failures. By parameter adjustments one can influence the routes, traffic splitting, and scheduling, and thus the performance of the network can be improved.

The IP routing paradigm provides services only on the best-effort basis. However, due to different needs of different customers, the need to differentiate services is evident in the Internet. QoS mechanisms, such as Integrated Services (IntServ) and Differentiated Services (DiffServ), have been widely discussed in recent years. The problem of IntServ is poor scalability, whereas the level of service differentiation provided by DiffServ is hard to predict. One possibility in providing tools for QoS in the Internet is the use of Multi Protocol Label Switching (MPLS) and its traffic engineering capabilities. Differentiation can be provided by balancing the load in such a way that different classes are routed separately.

In wireless mesh networks (WMNs) the use of load balancing approaches is also necessary. In a balanced network the delays are more tolerable and the robustness of the network against sudden changes in traffic is better. However, as compared to wired IP networks, the load balancing problem

is more complicated because of interference. Indeed, the link capacities are not fixed but may be changed by scheduling within certain limits. The joint planning of resource sharing on both the spatial and temporal levels is important for WMNs.

In this chapter we study load balancing in different types of networks. We start by formulating the standard static load balancing problem, also known as the multi-commodity flow problem. Then a simple adaptive algorithm to solve this problem is proposed and its applications in both MPLS and OSPF networks are studied. Finally, the static load balancing problem is examined for providing differentiated services in MPLS networks and optimal performance in wireless mesh networks.

3.2 Static load balancing problem

In this section we start the study of traffic engineering by formulating a classic load balancing problem and introducing different objective functions. In this problem the offered traffic of each ingress-egress (IE) pair is carried in the network in such a way that some objective function is minimized. Depending on the application, the paths used by each IE pair can be either entirely arbitrary or constrained to belong to a predefined set of alternative paths. We also review one algorithm, namely the gradient-projection algorithm, for solving the static load balancing problem.

To make a difference between names OD and IE pair, ingress and egress nodes refer to edge nodes of a routing domain, whereas origin and destination nodes refer to source and destination hosts defined by the IP address.

Undefined path set

Let us consider a single domain of an IP network, which consists of nodes and links connecting them. Let $\mathcal{N}$ denote the set of nodes (routers) n and $\mathcal{L}$ the set of links l of the network. Alternatively we use notation (i, j) for a link from node i to node j. The capacity of link l is denoted by b_l. The set of ingress-egress (IE) pairs $k = (s_k, t_k)$ is denoted by $\mathcal{K}$ with s_k referring to the ingress node and t_k referring to the egress node of IE pair k. The traffic demand, that is, the mean rate of offered traffic between nodes s_k and t_k, is denoted by d_k.

Furthermore, $A \in \mathbb{R}^{N\times L}$, where $N = |\mathcal{N}|$ and $L = |\mathcal{L}|$, denotes the link-node incidence matrix for which $A_{nl} = -1$ if link l directs to node n, $A_{nl} = 1$ if link l leaves from node n, and $A_{nl} = 0$ otherwise; and $R^k \in \mathbb{R}^{N\times 1}$, $k \in \mathcal{K}$, denotes the node-demand vector for which $R^k_{s_k} = d_k$, $R^k_{t_k} = -d_k$, and $R^k_n = 0$ otherwise.

The rate of traffic allocated by IE pair k on link l is denoted by x^k_l. These rates are the control variables in our static load balancing problem. Let $x^k \in \mathbb{R}^{L\times 1}$ denote the corresponding link load vector for IE pair k. Given the x^k_l, the induced load y_l on link l is

$$y_l = \sum_{k\in\mathcal{K}} x^k_l, \quad \text{for all } l \in \mathcal{L}.$$

Load y_l on link l incurs cost $C_l(y_l)$, which is assumed to be an increasing and convex function of link load y_l.

The objective in the *unconstrained static load balancing problem* is to minimize the total cost $C(x) = \sum_{l \in \mathcal{L}} C_l(y_l)$ by choosing an optimal traffic allocation $x^* = (x_l^k; k \in \mathcal{K}, l \in \mathcal{L})$. The problem is formulated as follows:

$$\begin{aligned} &\text{Minimize} \quad C(x) = \textstyle\sum_{l \in \mathcal{L}} C_l(y_l) \\ &\text{subject to the constraints} \\ &x_l^k \geq 0, && \text{for each } l \in \mathcal{L} \text{ and } k \in \mathcal{K} \\ &y_l \leq b_l, && \text{for each } l \in \mathcal{L}, \\ &Ax^k = R^k, && \text{for each } k \in \mathcal{K}, \end{aligned} \tag{3.1}$$

In this problem we have three constraints, the first one states that link loads should be non-negative, the second one is the capacity constraint and the third one is so called *conservation of flow constraint*, which states that the traffic of each IE pair incoming to a node has to be equal to the outgoing traffic from that node.

As a result of problem (3.1), linkwise traffic loads x_l^k that are fractions of the original demands d_k, are obtained. The set of routes computed from these loads for each demand through the network is not necessarily unique. However, the routes are guaranteed to be loop-free.

Predefined path set

Now we consider the case where the available paths are predefined. Each IE pair k has a predefined set $\mathcal{P}_k$ of paths. Let $\mathcal{P} = \cup_{k \in \mathcal{K}} \mathcal{P}_k$ denote the set of all paths. Each path $p \in \mathcal{P}_k$ consists of a concatenation of consecutive links from s_k to t_k. We use notation $l \in p$ whenever link l belongs to path p.

Let x_p denote the rate of traffic allocated to path p. Instead of the link loads of the previous case, the pathwise loads x_p are the control variables in the static load balancing problem with the predefined path set satisfying

$$\sum_{p \in \mathcal{P}_k} x_p = d_k, \quad \text{for all } k \in \mathcal{K}.$$

Given the x_p, the induced load y_l on link l is

$$y_l = \sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}_k : l \in p} x_p, \quad \text{for all } l \in \mathcal{L}.$$

The objective in the *constrained static load balancing problem* is to minimize the total cost by choosing an optimal traffic allocation $x^* =$

$(x_p^*; p \in \mathcal{P})$. The problem is formulated as follows:

$$\begin{aligned} &\text{Minimize} \quad C(x) = \textstyle\sum_{l \in \mathcal{L}} C_l(y_l) \\ &\text{subject to constraints} \\ &x_p \geq 0, && \text{for all } p \in \mathcal{P}, \\ &y_l \leq b_l, && \text{for all } l \in \mathcal{L}, \\ &\textstyle\sum_{p \in P_k} x_p = d_k, && \text{for all } k \in \mathcal{K}. \end{aligned} \tag{3.2}$$

As in problem (3.1), the pathwise solution is not necessarily unique when the paths share common links. If the path set $\mathcal{P}$ contains all possible paths between the ingress and egress nodes, the constrained static load balancing problem is equivalent to problem (3.1).

Optimization objectives

The static load balancing problem can be solved in terms of various objective functions. The functions can be categorized to linear and non-linear, resulting in linear optimization programs (LP) and non-linear optimization programs (NLP), respectively. An assumption is that the objective function is convex and increasing in its input parameters.

The simplest optimization problem in the networks is so called *minimum cost flow problem* [Ber98]. In that problem, each link l has unit cost a_l. The linkwise cost function is then

$$C_l(y_l) = a_l y_l, \quad \text{for all } l \in \mathcal{L}. \tag{3.3}$$

If the cost weights are selected to be $a_l = 1$ for all l, the optimization problem minimizes the total amount of used resources. On the other hand, if only one path is allowed to each IE pair, the problem is equivalent to the *shortest path problem*.

Another frequently used link cost is the following nonlinear function:

$$C_l(y_l) = \begin{cases} \dfrac{y_l}{b_l - y_l}, & y_l < b_l, \\ \infty, & y_l \geq b_l. \end{cases} \tag{3.4}$$

With this choice, the total cost function $C(x)$ tells the expected number of packets in the network at the moment. The mean delay of the network is the expected number of packets divided by the total packet arrival rate Λ. The required assumption is that each output buffer of a node can be modelled as an M/M/1 queue and the network as a Jackson queueing network. Although this assumption is not necessarily valid in a real network the objective function is useful in congestion reduction since the cost increases nonlinearly to infinity as the traffic load approaches the capacity of the link [Ber98]. In some contexts, as in [FT00], optimization of a non-linear problem is considered too time-consuming and thus it is approximated with piecewise linear optimization which can be solved optimally in polynomial time.

A third common objective function in load balancing is minimization of maximum link cost instead of the total cost. The problems (3.1) or (3.2) are replaced by following:

$$\begin{aligned} &\text{Minimize} \quad C(x) = \max_{l \in \mathcal{L}} C_l(y_l) \\ &\text{subject to constrains as in (3.1) or (3.2).} \end{aligned} \tag{3.5}$$

In this case a typical cost function is

$$C_l(y_l) = \frac{y_l}{b_l}. \tag{3.6}$$

By this choice the maximum link utilization is minimized. In the linear program formulation of the problem, instead of directly minimizing the maximum link utilization, the objective is to minimize free parameter α so that only fraction α of the bandwidth of each link can be utilized. The capacity constraint is thus

$$\sum_{k \in \mathcal{K}} x_l^k \leq \alpha b_l,$$

and the corresponding LP problem for the arbitrary path set is:

$$\begin{aligned} &\text{Minimize } \alpha \\ &\text{subject to the constraints} \\ &x_l^k \geq 0, && \text{for each } k \in \mathcal{K} \text{ and } l \in \mathcal{L}, \\ &\sum_{k \in \mathcal{K}} x_l^k \leq \alpha b_l, && \text{for each } l \in \mathcal{L}, \\ &Ax^k = R^k, && \text{for each } k \in \mathcal{K}. \end{aligned} \tag{3.7}$$

By problem formulation (3.7) we can find a unique solution for α, but the load allocation is not necessarily unique leaving room for satisfying some secondary criteria. As regards to such secondary optimization objective, some solutions are probably better than others. For example, if there is a bottleneck in the network, the rest of the network can remain unbalanced. The paper [OBC+01] defines that a routing that allocates traffic load y_l on link l is *non-dominated* if there does not exist another routing with link-loads y'_l with property

$$y'_l \leq y_l \text{ for all } l \text{ and } y'_l < y_l \text{ for at least one } l. \tag{3.8}$$

In other words, non-dominance means that it is not possible to find another optimal solution that does not use more capacity on any link and uses less capacity at least on one link. The problem of minimization of maximum link utilization (3.5) and (3.6) has the disadvantage that an optimal solution is not necessarily non-dominated. For this reason, after finding the minimum of the maximal link utilization α^* by optimizing problem (3.7), the traffic allocation should be optimized again in the network where the link

capacities are reduced by the factor α^*. The objective function in the second phase can be, e.g., minimization of the used resources (3.3). Another choice for the two-step optimization is the use of a combined parametric criterion as done in [OBC+01] and [WWZ01].

In the case where the path set is predefined the optimization problem for minimizing the maximum link utilization reads:

$$\begin{aligned}
&\text{Minimize } \alpha \\
&\text{subject to the constraints} \\
&\alpha \geq 0, \\
&x_p \geq 0, && \text{for all } p \in \mathcal{P}, \\
&\textstyle\sum_{k\in\mathcal{K}} \sum_{p\in P_k : l\in p} x_p \leq \alpha b_l, && \text{for all } l \in \mathcal{L}, \\
&\textstyle\sum_{p\in P_k} x_p = d_k, && \text{for all } k \in \mathcal{K}.
\end{aligned} \tag{3.9}$$

Solving the optimization problem

There is a wide range of different algorithms to solve the static load balancing problem, starting from the simplex method for linear problems. Best-known algorithms to optimize non-linear problems are the steepest descent, conjugate gradient, Newton's and gradient projection methods. In this section we review the gradient projection method in more detail because of its suitability for distributed computation in communication networks.

A gradient-projection algorithm for the static load balancing problem with a predefined path set is presented in [TB86]. In this algorithm, one first calculates the cost C_p for each path $p \in \mathcal{P}_k$, which is a sum of the link costs along path p:

$$C_p = \sum_{l\in p} C_l(y_l). \tag{3.10}$$

The first derivative length of this path is

$$C'_p = \sum_{l\in p} C'_l(y_l), \tag{3.11}$$

where $C'_l(y_l)$ is the first derivative of the cost function $C_l(y_l)$ and y_l is the induced traffic load on link l (see Section 3.2).

Let $\bar{p}_k$ denote the path of the IE pair k which has the smallest first derivative length (3.11). In the algorithm the traffic load is iteratively moved toward the direction of the gradient projection of the objective function. Therefore at each iteration $t+1$, the traffic allocation of path p is updated as follows:

$$x_p(t+1) = \max[x_p(t) + \frac{\alpha_H}{H_p}(C'_{\bar{p}_k} - C'_p), 0] \quad \forall p \in P_k, p \neq \bar{p}_k,$$

where α_H and H_p are scaling factors. Factor α_H is selected freely, but factor H_p is defined as:

$$H_p = \sum_{l\in p\cup\bar{p}_k, l\notin p\cap\bar{p}_k} C''_l(y_l),$$

where $C_l''(y_l)$ is the second derivative of cost function $C_l(y_l)$ with respect to link load y_l.

To ensure that the new solution is feasible, that is, the sum of the traffic allocations on the paths is equal to the traffic demand d_k, the extra traffic of each IE pair is allocated to the path with the shortest first derivative length, that is,

$$x_{\bar{p}_k}(t+1) = d_k - \sum_{p \in P_k, p \neq \bar{p_k}} x_p(t+1).$$

3.3 Related research

TE in OSPF networks

Recent modifications to OSPF routing to support traffic engineering are presented in RFC 3630 [DKY03]. The idea of TE extensions is to add more parameters describing the state of a link to the link state advertisements and thus extend the link state database.

One possibility for optimizing the performance of an OSPF network is to tune the link weights and thus the shortest paths. The earliest proposal to tune OSPF-weights to achieve an optimal routing was presented by B. Fortz and M. Thorup in [FT00]. The assumption in the paper is that traffic is split equally among the adjacent links that are on the shortest paths to the destination. The result of the paper is that in general it is not possible to select link weights so that problem (3.1) is optimized. Even finding a best weight setting is an NP-hard problem. In [FT02] the same authors point out that weight changes should be avoided as much as possible. Finally, in [MRZ03] and in [AC03] the OSPF weight optimization is studied without exact knowledge of the traffic matrix and in [ERP] and in [BRRT02] genetic algorithms are presented for finding a good OSPF-weight setting.

Also Wang et al. [WWZ01] and Sridharan et al. [SGD03] investigate the weight setting problem. However, the assumption concerning the traffic splitting to multiple shortest paths differs fundamentally from the earlier papers; instead of equal splitting, arbitrary splitting ratios are allowed. The important result of [WWZ01] is that optimal routing in terms of any objective function can be converted to the shortest-path routing with the positive link weight set if traffic can be split arbitrarily to the shortest paths and each OD pair is forwarded distinctly. First the paper formulates a linear optimization problem with the optimal traffic allocation of the links as side constraints. The link weights are then found as a solution to the dual of the linear problem.

Research concerning adaptive approaches to optimize OSPF networks has not been very intensive. Many mechanisms have been developed to improve the performance of the network by changing the link weights and thus the shortest paths adaptively (see [PL02] and [KYKV03]), but this can lead to undesired effects in the network performance and is technically difficult. Examples of adaptive mechanisms that rely on traffic distribution on multiple shortest paths are OSPF-OMP and AMP. OSPF Optimized Multipath (OSPF-OMP) described in [Vil99] is a mechanism that extends OSPF to support traffic-aware routing by forwarding traffic to the shortest paths unequally based on measurements. The purpose of Adaptive Multi-

Path (AMP) algorithm proposed in [GZRR03] is to reduce the memory consumption and signalling overhead of OSPF-OMP algorithm using only local information in optimization.

A drawback of TE methods reviewed in this subsection is that the traffic demand matrix is assumed to be known. In addition, the approaches do not take the traffic fluctuations into account. The few adaptive approaches presented in the literature do not provide a proper comparison of the approaches to min-hop routing and optimality, and the analysis of the stability issues is missing.

TE in MPLS networks

There is a lot of research related to traffic engineering in MPLS networks starting from static optimization techniques and adaptive algorithms and ending with TE with service differentiation.

First Ott et al. [OBC+01] consider a static non-linear optimization problem, such as minimization of the mean delay. Since solving the NLP problem requires a lot of computational effort, they present a two-step algorithm, in which paths are calculated by LP-optimization and then traffic is allocated to paths using NLP-optimization. The granularity aspects of traffic engineering are studied in [SBG+02], where simple heuristics for load balancing using a particular level of traffic granularity is developed.

Dinan et al. [DAJ00] study analytically how the traffic can be mapped to multiple parallel paths in an optimal way. In the study, LSPs are modelled as sequences of M/M/1/K-queues building up a queueing network. In the algorithm, each ingress node defines its traffic mapping vector to parallel LSPs based on the network state in such a way that the loss rate is minimized. Also Zhao et al. propose a queueing-network model for load balancing in [ZSZ+04]. In the work each LSP is modelled as an M/G/1/PS-queue.

Elwalid et al. [EJLW01] introduce a mechanism called MPLS Adaptive Traffic Engineering (MATE). MATE is a state-dependent traffic engineering mechanism, in which the traffic load is balanced using a distributed adaptive algorithm. The delay and packet loss are measured periodically by sending probe-packets. When a change of predefined size in these traffic conditions is perceived, the algorithm moves to the load balancing phase and tries to equalize congestion measures among the LSPs by approximating the gradient-projection algorithm (introduced in Subsection 3.2). In paper [GKL+04] some criticism concerning the MATE mechanism is raised. The problem of MATE is that, although it is said that the cost derivatives cannot be computed, it assumes the existence of the analytical gradient function in the algorithm. To overcome this problem [GKL+04] proposes some enhancements to MATE.

Song et al. [SKL+03] introduce a concept called Load Distribution over Multipath (LDM), in which traffic is split dynamically at flow level into multiple paths. The set of available LSPs is fixed.

A mixed integer programming (MIP) problem for hop-count and path-count constrained multipath routing is presented in [SLC01]. Since the problem is NP-hard, the paper proposes heuristics to find multiple paths and their split ratios for each traffic demand request between the ingress

and the egress node.

Butenweg proposes in [But03] two distributed and reactive load balancing mechanisms. The first one is based on rerouting of Label Switched Paths (LSP) and the second on multi-path load balancing. As the LSR detects an overloaded link, it chooses an LSP for rebalancing. In the rerouting approach the source LSR of the selected LSP computes a new path for the LSP. In the multi-path approach, instead of rerouting the LSP, the source LSR adjusts the weights which define the distribution of traffic to parallel LSPs.

We find some drawbacks in the algorithms discussed in this subsection. First, in [EJLW01] traffic engineering is based on the measured traffic conditions between each ingress and egress pair. The use of active measurements loads the network, and moreover, these measurements offer redundant information since LSPs might use same links. The amount of control data can be reduced by using only link load information. Second, momentary efficiency of the algorithm in [SKL+03] depends largely on the flow-level dynamics. It remains unclear whether the granularity of the traffic splitting at the flow level is fine enough to provide stable network conditions.

TE with differentiated services

Although Quality of Service is a hot issue in today's Internet development, the first versions of MPLS do not provide QoS capabilities directly [Spr00]. There is no mechanism to send all the high-priority traffic to the destination using a particular path and then send best-effort traffic to the destination using a different path.

One proposal to achieve QoS is a Provider Architecture for Differentiated Services and Traffic Engineering (PASTE) [LR98]. PASTE utilizes MPLS and RSVP to provide scalable traffic management architecture. Another possibility to achieve Quality of Service is an infrastructure introduced in [FWD+02]. The structure specifies a solution that supports Behavior Aggregates (BA) of Differentiated Services in an MPLS domain. The solution presented in [FWD+02] includes two types of LSPs for different traffic classes. Finally, Internet Draft [FTT+02] introduces Service Provider requirements that enable DiffServ-aware MPLS Traffic Engineering (DS-TE). In the DS-TE model traffic engineering is done at a per-class level instead of the aggregate level. Traffic of a particular class can be mapped to separate LSPs so that the resources of a network are utilized.

QoS-based routing is a routing mechanism, where paths for a particular flow are selected based on the information of available resources in the network and QoS requirements of flows [CNRS98]. The three major elements of QoS-based routing according to [AWK+99] are 1) acquiring the required information and determining a feasible path based on QoS requirements of a given request, 2) accommodation of a new request by establishing the path selected and 3) the maintenance of the selected path.

Wireless mesh network optimization

During the last decade multi-hop wireless networks have received much research effort. Wireless mesh networks (WMNs) cover a wide range of

different radio technologies and thus the number of performance optimization challenges is huge. In this subsection we introduce some examples of research related to WMNs, concentrating on the work in which the MAC layer of the wireless network is modelled by Spatial TDMA [NK85].

First Gupta and Kumar present in [GK00] a study where they show that in a wireless network with n nodes, the throughput per node can be calculated as $\Theta(1/\sqrt{n \log n})$ if the nodes are placed randomly, and $\Theta(1/\sqrt{n})$ if the placement of the nodes as well as communication pattern is optimized.

Björklund et al. [BVY03] study the optimal allocation of the time slots to a given set of the nodes or links. The interference constraints of the STDMA model are included in the optimization problem resulting in a computationally complex system. To solve this system effectively, different approaches are proposed in, e.g., [BVY03] and [JX06].

The objective of the work of Jain et al. in [JPPQ05] is to maximize the throughput of the network with a given set of the nodes and traffic patterns. The paper proposes a *conflict graph* to present the interference between neighboring nodes and computes lower and upper bounds for optimal throughput. Wu et al. [WCZ+05] consider the cross-layer optimization of network resources for multicast sessions in such a way that a congestion measure is minimized.

Finally, Lassila et al. [LPV06] consider the cross-layer dimensioning of WMNs. The objective is to dimension the nominal capacity of the wireless links so that certain QoS level requirements are fulfilled.

3.4 Contribution

In this thesis we study load balancing in the Internet by presenting a novel adaptive algorithm for improving the performance of the network. The implementation of the algorithm in MPLS as well as in OSPF networks is considered. In addition, we formulate the load balancing problem in the networks with service differentiation as well as in wireless mesh networks. Detailed contributions of the papers are provided next.

Publication 4 proposes a simple adaptive and distributed load balancing algorithm that makes incremental changes in splitting the traffic to multiple parallel paths in MPLS networks. The adjustments are done based on the measured link loads without information about the end-to-end traffic demands. The algorithm is tested by a numerical evaluation method and it is found that an almost optimal traffic allocation of the network can be achieved in a distributed way, if the step size of the algorithm is sufficiently small.

Publication 5 extends the algorithm of Publication 4 to OSPF networks. Whereas in MPLS networks the packet is forwarded through predefined paths from the ingress node to the egress node, in OSPF networks each router independently makes routing decisions. For that reason also load balancing is done at each router for the aggregate traffic independent of the origin node of the incoming packet. Results show that also by this approach optimal load distribution can be achieved, even if traffic fluctuates randomly.

Publication 6 studies how MPLS and load balancing can be used for

achieving different service levels for different traffic classes. Two different approaches are considered: differentiation based on the routing and differentiation based on the routing and scheduling. Static optimization problems for these approaches are formulated and also many approximative algorithms given.

Finally, Publication 7 studies load balancing in wireless multihop network. A static load balancing problem for optimizing the traffic routing and scheduling is formulated for both undefined and predefined path sets. By numerical examples, the optimal way to share the resources is compared to other approaches and found that only by joint optimization of routing and scheduling improvements in congestion can be obtained.

In the next three sections we review the problem formulations of the papers as well as present the main achievements in more detail. First we focus on the adaptive load balancing in MPLS and OSPF networks and then on load balancing with differentiated services and in wireless multihop networks.

3.5 Adaptive load balancing in MPLS and OSPF networks

In this section we suggest a simple adaptive and distributed load balancing algorithm that continuously makes incremental changes in the load distribution based on measured link loads. The changes are made by the specific routers independently of each other. No prior information about the traffic demands is required. The algorithm also adapts itself to changing demands.

Proposal of an adaptive load balancing algorithm

Consider a network consisting of routers $n \in \mathcal{N}$ and links $l \in \mathcal{L}$. In addition, we have a predefined set $\mathcal{P}$ of paths. The path p between given ingress and egress nodes can be considered to belong to same sub-group r of paths, denoted by $\mathcal{P}_r$. Selection of these paths depends on the underlying routing protocol (MPLS or OSPF in our work) and will be defined later. In our adaptive load balancing algorithm the ingress router of each path set $\mathcal{P}_r$ is responsible for making the load balancing decisions. These nodes are called as *LB routers*.

The network is loaded by the set of traffic demands $\mathcal{K}$ in such a way that traffic of demand k goes from ingress node s_k to egress node t_k. In our scenario these demands are unknown. Instead, we assume that the link loads are measured periodically at times t_n. These measured link loads $\hat{y}_l(n)$ should be understood as time-averages over the whole measurement period (t_{n-1}, t_n), and not as instantaneous values. On the other hand, due to traffic fluctuations of demands on different time scales, the measured loads also deviate from the long-term time averages in a random manner (we will later assess the effect of the traffic fluctuations by numerical experiments).

In the end of each measurement period we assume that the information about the loads is distributed to all LB routers in the network. This can be done in a similar way as the link states are distributed to all routers within an AS in OSPF [Moy98]. We further assume that the time needed to distribute the information to edge routers is negligible in comparison to the length of the measurement period. In the OSPF flooding protocol, the delay to

distribute the link state advertisements is principally the same as the end-to-end network delay (many papers use 5 seconds as a reference value).

Let $\phi_p(n)$ denote the *proportion* of traffic allocated to path p at time t_n in path set $\mathcal{P}_r$. These splitting ratios are the control variables in our dynamic load balancing problem satisfying, for all r and n,

$$\sum_{p \in \mathcal{P}_r} \phi_p(n) = 1.$$

In the algorithm splitting ratios $\phi_p(n)$ are gradually adjusted based on the measured link loads $\hat{y}_l(n)$ resulting in an *adaptive* load balancing algorithm. It is also important to note that these splitting ratios determine the proportions of the incoming traffic, not the absolute amounts. This kind of traffic splitting may be done at the packet level or at the flow level. If traffic splitting is done at the packet level, reordering of packets may be required. Splitting traffic at the flow level, as in [SKL+03], avoids this problem, but then poor and unpredictable granularity may be a drawback.

Assuming that the traffic demands are constant (albeit unknown), the objective in the *dynamic load balancing problem* is to choose, after each measurement period (t_{n-1}, t_n), the splitting ratios $\phi(n) = (\phi_p(n); p \in \mathcal{P})$ in such a way that they converge, as quickly as possible, to the unknown optimal values ϕ_p^* of the corresponding static load balancing problem, e.g. (3.2).

Since the measured loads $\hat{y}_l(n)$ are distributed to all LB routers, the decisions concerning the splitting ratios $\phi_p(n)$ can be made in a distributed way. Each LB router independently determines the splitting ratios $\phi_p(n)$ for all those paths p for which it is the ingress node.

The adjustment of the splitting ratios will depend on the estimated path costs $D_p(\hat{y}(n))$, where $\hat{y}(n) = (\hat{y}_l(n); l \in \mathcal{L})$ denotes the vector of measured loads. The definition of the path cost $D_p(\hat{y}(n))$ depends on the formulation of the original optimization problem. If the purpose is to minimize the total cost (3.2), such as the mean delay, then we define two different path costs. One is the total cost along the path,

$$D_p(\hat{y}(n)) = \sum_{l \in p} C_l(\hat{y}_l(n)), \tag{3.12}$$

and the other one is its first derivative length,

$$D_p(\hat{y}(n)) = \sum_{l \in p} C_l'(\hat{y}_l(n)). \tag{3.13}$$

But if the objective is to minimize the maximum cost (3.5), such as the maximum utilization, then the path costs are naturally defined by

$$D_p(\hat{y}(n)) = \max_{l \in p} C_l(\hat{y}_l(n)). \tag{3.14}$$

The idea is simply to alleviate the congestion on the most costly path (among the paths belonging to same path set $\mathcal{P}_r$) by reducing its splitting ratio. This should, of course, be compensated for by increasing the splitting ratio of some other path within the same set $\mathcal{P}_r$. We study two different rules: (a)

choose the other path randomly among the other paths, or (b) choose a path with minimum path cost.

Since the algorithm is adaptive, we have a closed-loop control problem: the splitting ratios that depend on measured loads have a major effect on the upcoming load measurements. To avoid harmful oscillations, we let the splitting ratios change only with minor steps defined by granularity parameter g. A finer granularity is achieved by increasing the value of g.

Algorithm for determining the splitting ratios. At time t_n, after receiving the information concerning all the measured loads $\hat{y}_l(n)$, the LB router operates for all path sets $\mathcal{P}_r$ for which it is an ingress router, as follows:

1. Calculate the path costs $D_p(\hat{y}(n))$ for each path $p \in \mathcal{P}_r$.
2. Find the path $q \in \mathcal{P}_r$ with maximum cost, i.e.

$$D_q(\hat{y}(n)) = \max_{p \in \mathcal{P}_r} D_p(\hat{y}(n)),$$

   and decrease its splitting ratio as follows:

$$\phi_q(n) = \phi_q(n-1) - \frac{1}{g}\phi_q(n-1).$$

3. Choose another path $u \in \mathcal{P}_r$ either
   a) randomly, or
   b) so that the path cost is minimized, i.e.

$$D_u(\hat{y}(n)) = \min_{p \in \mathcal{P}_r} D_p(\hat{y}(n)),$$

   and increase its splitting ratio as follows:

$$\phi_u(n) = \phi_u(n-1) + \frac{1}{g}\phi_q(n-1).$$

4. For all other paths $p \in \mathcal{P}_r$, keep the old splitting ratios,

$$\phi_p(n) = \phi_p(n-1).$$

Application of the algorithm to MPLS networks

In MPLS networks, traffic of demand k is routed from ingress node s_k to egress node t_k along predefined set of paths (LSPs), denoted by $\mathcal{P}_k$. The application of the adaptive load balancing algorithm to MPLS networks is thus straightforward. The path set $\mathcal{P}_r$ between two nodes introduced in the previous subsection is now replaced by LSPs $\mathcal{P}_k$ set up for specific demand k. The LB router of the adaptive load balancing algorithm is indeed the origin router of the LSP, s_k, for each IE pair k and the decisions to adjust the splitting ratios are done separately at these routers. See Figure 3.1 for an illustration.

Next we evaluate the performance of the proposed adaptive algorithm in MPLS networks. The evaluation method is iterative and runs as follows.

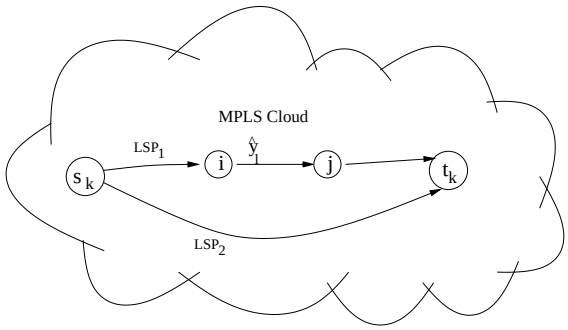

Figure 3.1: The network model

The test network is loaded by the traffic demands d_k between nodes s_k and t_k. Note that the traffic demands are used only for the evaluation of the proposed load balancing algorithm, but the algorithm itself does not require information about them. The traffic loads of the IE pairs fluctuate around the deterministic value d_k during measurement period n, resulting in demands

$$\tilde{d}_k(n) = d_k + \epsilon_k(n), \quad k \in \mathcal{K}, \tag{3.15}$$

where the $\epsilon_k(n)$ are selected as independent Gaussian random variables with mean 0 and variance $\delta^2 d_k^2$.

At each iteration i, depending on the splitting ratios $\phi_p(n-1)$, i.e. routing, traffic demands induce the link loads:

$$\hat{y}_l(n) = \sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}_k : l \in p} \tilde{d}_k(n) \phi_p(n-1),$$

that are measured. The new splitting ratios $\phi_p(n)$ are determined from the measured loads $\hat{y}_l(n)$ by the adaptive algorithm presented in the previous subsection.

The test-network in this thesis consists of 10 nodes, 52 links, and 72 IE pairs. In the upper graph of Figure 3.2 the optimization objective is the minimization of the maximum link utilization (3.5). The path cost is the minimum of link loads (function (3.14)) and the new path for the shifted traffic is selected randomly (rule 3a in the algorithm). The effect of traffic fluctuations on measured link loads is ignored, that is, variation parameter $\delta = 0$. The traffic demands are such that the min-hop routing results in the maximum link utilization of 0.88, while the optimal value is 0.54. We can see that the values obtained from the adaptive algorithm after 100 iterations are close to the optimal ones. With a coarse granularity ($g = 20$), the convergence is naturally faster (about in 30 iterations) but small oscillations can be observed. With a finer granularity ($g = 50, 100$), the oscillations disappear almost completely.

In the lower graph of Figure 3.2 the objective is to minimize the mean delay. We study the following variants of our adaptive load balancing algorithm: (i) path cost (3.12) and rule 3a in the algorithm (denoted by RNDPTH), (ii) path cost (3.12) and rule 3b in the algorithm (MINCST)

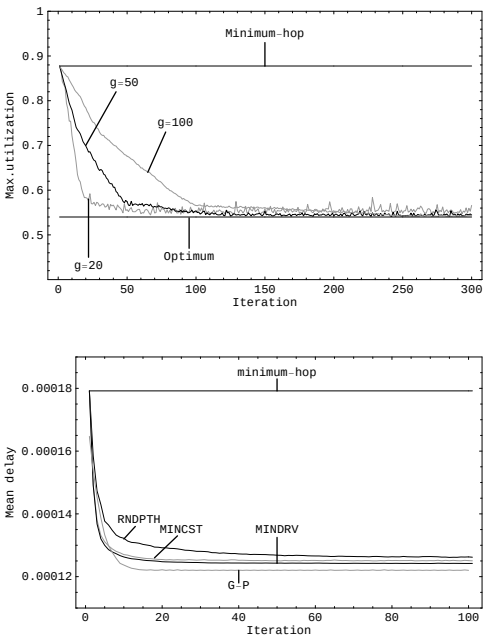

Figure 3.2: Upper graph: The maximum link utilization as a function of the number of iterations for different levels of granularity. Lower graph: The mean delay as a function of iterations with different variants of the algorithm, when $g = 10$.

and (iii) path cost (3.13) and rule 3b in the algorithm (MINDRV). In addition, we compare the results to the gradient-projection algorithm (GP), which requires information about the pathwise traffic loads. From the figure we can see that the convergence of the algorithm is achieved approximately in 20 iterations in all cases ($g = 10$). The performance of the simple algorithm does not differ significantly from that of the `G-P` algorithm. However, the number of updates per iteration in our adaptive algorithm is much smaller, since the splitting ratios of only two LSPs for each IE pair are updated whereas in the `G-P` algorithm all the splitting ratios are updated. In comparing different variants of the algorithm we note that `MINDRV` seems to converge closest and fastest to the optimal value.

Application of the algorithm to OSPF Networks

Unlike in MPLS networks, in OSPF networks we cannot directly determine the paths and splitting ratios used by each end-to-end traffic demand. Instead, the routing decisions for the data are made at each router separately independent of the source of the data.

When OSPF routing is considered, each link l is associated with a fixed weight w_l and the traffic is carried along shortest paths. Let $\mathcal{P}_k^{\mathrm{SP}}$ denote the set of shortest paths for IE pair k and $\mathcal{P}_{i,t}^{\mathrm{SP}}$ shortest paths from node i to

node t with respect to link weights w_l.

At each node i, the incoming traffic with the same destination t is aggregated and then split to links (i,j) that belong to some of the shortest paths of set $\mathcal{P}^{\text{SP}}_{i,t}$. Such adjacent nodes j are called admissible next hops. Let ϕ^t_{ij} denote the corresponding splitting ratios. Thus, ϕ^t_{ij} refers to the fraction of overall traffic passing node i and destined to node t that is forwarded on link (i,j). It is required that

$$\sum_{j:(i,j)\in p \text{ for some } p\in\mathcal{P}^{\text{SP}}_{(i,t)}} \phi^t_{ij} = 1.$$

For an illustration of the model, see Figure 3.3. By setting the splitting ratios ϕ^t_{ij} to equal values (independent of j):

$$\phi^t_{ij} = \frac{1}{|\{j' : (i,j') \in p \text{ for some } p \in \mathcal{P}^{\text{SP}}_{(i,t)}\}|},$$

the resulting routing follows the ECMP principle described earlier in Section 1.4. However, there are methods that allow unequal splitting ratios, such as explained in [Vil99] and [SGD03].

When our adaptive load balancing algorithm is implemented in an OSPF network, instead of considering end-to-end paths for each demand k, path-set P_r is replaced by the shortest path set $\mathcal{P}^{\text{SP}}_{(i,t)}$ for each node pair (i,t). In addition, the splitting ratios are not determined for each path separately, but for next hops of the paths according to variables ϕ^t_{ij} explained above. Note that for each path considered, the corresponding next hop and splitting ratio is determined explicitly. As a result of implementation of the algorithm, all the nodes making routing decisions in the network make also load balancing decisions. Recall also that the decisions do not depend on the ingress node of the incoming traffic as in the MPLS case, but are made for the aggregate traffic.

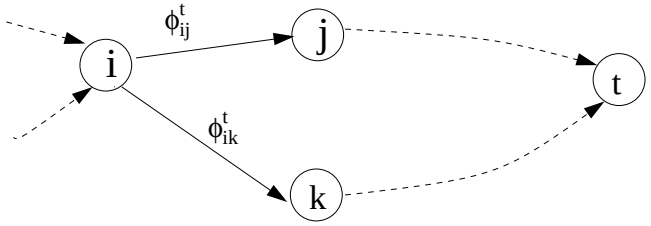

Figure 3.3: The OSPF network model

Evaluation of the adaptive algorithm in OSPF network is similar to the MPLS network case. Traffic demands are initially allocated to the shortest paths with respect to the fixed link weights w_l. If multiple shortest paths exist, traffic is initially split equally in each node (ECMP). At each iteration n, the measured link loads $\hat{y}_l(n)$ induced by the splitting ratios $\phi^t_{ij}(n-1)$ are calculated as follows. First we calculate, for each IE pair k, the induced traffic splitting ratios $\phi_p(n-1)$ for each path $p \in \mathcal{P}^{\text{SP}}_k$ by

$$\phi_p(n-1) = \prod_{(i,j)\in p} \phi^{t_k}_{ij}(n-1).$$

Then the measured link loads $\hat{y}_l(n)$ are determined by

$$\hat{y}_l(n) = \sum_{k \in \mathcal{K}} \tilde{d}_k(n) \sum_{p \in \mathcal{P}_k^{\mathrm{SP}}: l \in p} \phi_p(n-1),$$

where $\tilde{d}_k(n)$ is defined by formula (3.15). New traffic splitting ratios $\phi_{ij}^t(n)$ are determined from the measured loads $\hat{y}_{(i,j)}(n)$ by the adaptive algorithm.

In the numerical examples our aim is to minimize the maximum link utilization.The results of the adaptive algorithm are compared with those of "ECMP", "Optimal", and "Sub-optimal". ECMP refers to the standard policy where the traffic is split equally to the shortest paths with the unit link weights. Instead, optimum is a result of optimization problem (3.7) with undefined path set. To achieve this optimal load allocation in OSPF networks, in addition to splitting ratios, also the link weights w_l have to be adjusted. The link weights can be found by a primal-dual method [WWZ01]: Let $\tilde{y}_l = \sum_k \tilde{x}_l^k$ denote the traffic load allocated to link l in the optimal solution $\tilde{x}_l^k$ of the load balancing problem (3.7). Formulate then another LP-problem (primal) and its dual. The primal LP-problem is the standard load balancing problem (3.1) with the total load (3.3) as an optimization objective and the induced link loads $\tilde{y}_l$ serving as new capacity constraints:

$$x_l^k \le \tilde{y}_l \quad \text{for each } l \in \mathcal{L}\ .$$

The link weights w_l can be found as a solution to the dual of the primal problem:

$$\begin{aligned}
&\text{Maximize } \sum_{k \in \mathcal{K}} d_k U_{t_k}^k - \sum_{l \in \mathcal{L}} \tilde{y}_l W_l \\
&\text{subject to the constraints} \\
&W_l \ge 0 && \text{for all } l \in \mathcal{L}, \\
&U_{s_k}^k = 0, && \text{for all } k \in \mathcal{K}, \\
&U_j^k - U_i^k \le W_{(i,j)} + 1, && \text{for all } k \in \mathcal{K} \text{ and } (i,j) \in \mathcal{L}.
\end{aligned} \tag{3.16}$$

The required link weights are given by $w_l = W_l + 1$, where the variables W_l are determined as the solution to the dual problem.

In addition, optimal destination-based traffic splitting ratios ϕ_{ij}^t are determined from the link loads x_l^k of the solution of the primal problem. These splitting ratios are calculated as follows [SGD03]:

$$\phi_{ij}^t = \frac{\sum_{k: t_k = t} x_{(i,j)}^k}{\sum_{j': (i,j') \in \mathcal{L}} \sum_{k: t_k = t} x_{(i,j')}^k}. \tag{3.17}$$

Finally, the sub-optimal value refers to the optimal value of problem (3.9) with restricted path set including all shortest paths with unit link weights. As a solution of problem (3.9), we obtain path-wise splitting ratios x_p, which can be converted to destination-based splitting ratios as follows:

$$\phi_{ij}^{t} = \frac{\sum_{k:t_k=t} \sum_{p \in \mathcal{P}_k^{\mathrm{SP}}:(i,j)\in p} d_k \phi_p}{\sum_{j':(i,j')\in\mathcal{L}} \sum_{k:t_k=t} \sum_{p \in \mathcal{P}_k^{\mathrm{SP}}:(i,j')\in p} d_k \phi_p}. \tag{3.18}$$

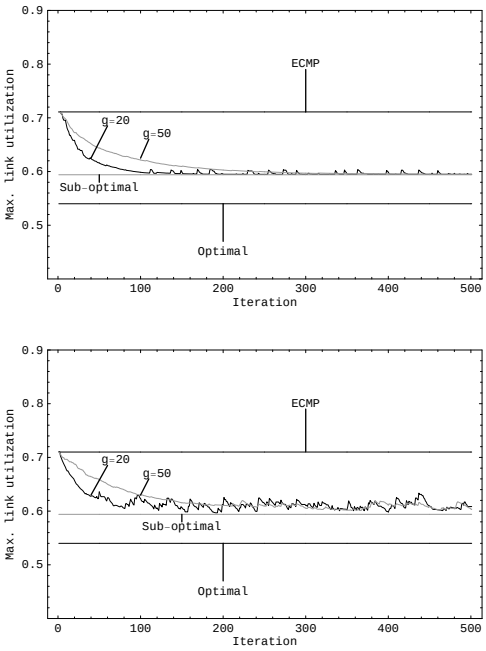

Figure 3.4: The maximum link utilization as a function of the number of iterations. Upper graph: $\delta = 0$, lower graph: $\delta = 0.1$.

Figure 3.4 shows the resulting maximum link utilization as a function of the number of iterations for granularity parameters $g = 20$ and $g = 50$. In the upper graph $\delta = 0$ and in the lower graph $\delta = 0.1$. We can see that the performance of the adaptive algorithm approaches the sub-optimal value and improves the performance remarkably as compared with the standard equal splitting. A small step size in the algorithm ensures that oscillations are insignificant. We find that the random traffic fluctuations on the time scale of the measurement period induce oscillations to the maximum link utilization but by a small step size this can be alleviated.

Comparison of the MPLS and OSPF load balancing

In last subsections we have studied adaptive load balancing in both connection-oriented MPLS networks and connectionless OSPF networks. Both technologies have their own benefits and limitations in performance optimization of the operational networks, which are discussed next.

In MPLS networks the support for the explicit paths is a part of technology. Those paths provide much flexibility to control the traffic traversing the network from the source to the destination. On the other hand, MPLS can bring out scalability problems in large networks, although scalability is better in comparison with the older overlay models, such as IP over ATM. The benefit of OSPF traffic engineering is that the shortest-path routing is the standard and most-used technology. However, balancing load without traffic splitting is hardly possible. If only a link weight is changed, it is difficult to forecast the effects caused to the network performance.

Next we compare the iteration times of the adaptive MPLS and OSPF algorithms. From the upper graph of Figure 3.2 we can see that approaching the close-optimal value in the minimization of maximum link utilization takes approximately 20 iterations when $g = 20$. Correspondingly, the upper graph of Figure 3.4 shows that the sub-optimum is achieved in an OSPF network in 100 iterations. The result is that the convergence times in MPLS networks are significantly shorter. The source-destination based decision parameters seem to work more effectively than the destination based parameters.

If we assume that the flooding of the link load information takes 5 seconds at least, the minimum time to convergence in MPLS networks is approximately 1.7 minutes, whereas it is in OSPF networks 8.3 minutes. If the measurement period is 5 minutes as typically in SNMP, the convergence time in MPLS network is approximately 1.7 hours and in OSPF networks 8.3 hours. However, it should be noted that the algorithm improves the performance of the network immediately after the first iterations, not only after the convergence to the optimum. Thus significant improvements in the load distribution can be achieved much faster than in 8.3 hours.

3.6 Proposal for differentiating service by load balancing in MPLS networks

In addition to the pure congestion management, MPLS and its load balancing functionalities provide means for differentiating service of different traffic classes. Differentiation can be achieved by routing the lower priority traffic to other routes than higher priority traffic. However, this approach provides only limited level of differentiation. To make the separation of different classes more effective, scheduling mechanisms, such as Weighted Fair Queueing (WFQ), can be used. The idea of WFQ is to provide a fair bandwidth distribution to the queues with different bandwidth requirements [Zha95, BZ97]. In contrast to the previous section, in this and the next section we consider load balancing as a static optimization problem that assumes knowledge of the end-to-end traffic demands.

Differentiation with routing

In this subsection we study load balancing by an approach that differentiates traffic classes based on the routing only. We consider a situation where the total traffic is divided into traffic classes, into the gold, silver and bronze classes, for example. The gold class has the highest priority and the bronze class the lowest priority.

The network is loaded by I traffic classes. Each traffic class has its own

traffic matrix and let $d_{i,k}$ denote the traffic demand of IE pair k and traffic class i. In addition, let $R_{i,k} \in \mathbb{R}^{N\times 1}$ denote the array for each class l and IE pair k, where $R_{i,k,n} = d_{i,k}$, if n is the ingress node, $R_{i,k,n} = -d_{i,k}$, if n is the egress node and $R_{i,k,n} = 0$ for all other nodes n. The total traffic offered by class i is denoted by Λ_i. Let $x_{i,k,l}$ be the allocated traffic of ingress-egress pair k of class i on link l. So the total amount of class i traffic on link l is

$$y_{i,l} = \textstyle\sum_k x_{i,k,l}, \quad \text{for all } i \text{ and } l. \tag{3.19}$$

The packet delay for class i is

$$E[D_i] = \frac{1}{\Lambda_i} \sum_l \frac{y_{i,l}}{b_l - y_{i,l}}.$$

Let w_i be the cost weight associated to traffic class i. Additional notation used is the same as in section 3.2.

The purpose in the optimization problem is to route traffic over paths so that classes with higher priority achieve smaller mean delay than other classes. The optimization function in the load balancing problem with service differentiation is thus:

$$C(x) = \sum_i w_i E[D_i].$$

When the differentiation is done by routing only, the constraints of the optimization problem are similar to the constraints of the standard load balancing problem (3.1). When the cost weights of different classes differ sufficiently, the optimization function tries to minimize the mean delays of gold class at the expense of lower priority classes. As a result, the routing obtained by the optimization function above differs from the load balanced routing, because in the load balancing the delays of the links are balanced to be almost equal and the differentiation could occur only if the paths are of different length.

Another possibility to achieve differentiation is to fix the ratio of the mean delays of various classes to some predefined value. By this approach certain level of differentiation can be achieved, but the optimization problem is more complicated.

A numerical example of the routing with service differentiation is shown in Figure 3.5. The test network is the same as in the previous section, except that the traffic of each demand is now divided into two classes. When the weight of the gold class is increased, the ratio of the mean delays increases (see the upper graph of Figure 3.5). On the other hand, the routing is not necessarily optimal and therefore the mean delay increases as the differentiation is greater, as can be seen from the lower of graph of Figure 3.5.

Differentiation with routing and scheduling

As mentioned earlier, possibilities to differentiate the traffic classes with routing are limited, and in addition, differentiation can lead to poor overall performance. Therefore in this subsection, we consider a scheme where,

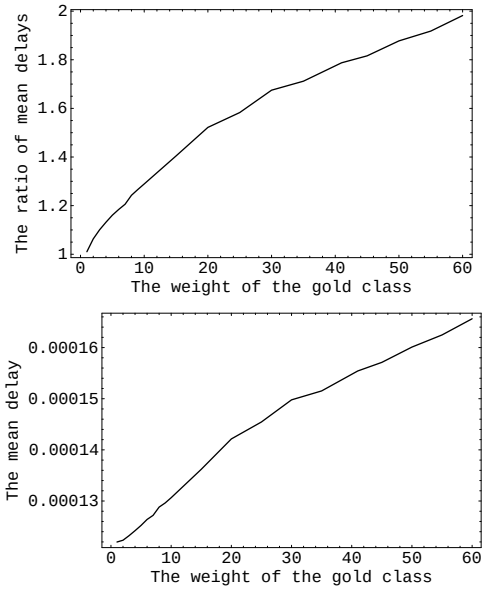

Figure 3.5: The ratio of mean delays (upper graph) and the total mean delay (lower graph) as a function of the weight of the gold class

in addition to routing, WFQ scheduling is used to achieve traffic differentiation. Each queue gets a guaranteed bandwidth, which depends on the WFQ-weight of that queue and the link capacity. We try to find optimal routing that differentiates the quality of service by including the WFQ-weighs to the optimization function as free parameters.

Because WFQ-scheduling is a work-conserving discipline, the bandwidth that is guaranteed for a class in our model can be viewed as the lower bound. Actually, the bandwidth available to a class can be much greater as the other classes may not always use the bandwidth reserved for them.

We assume that the bandwidth of each link l is shared according to the WFQ-weights. We approximate the behavior of WFQ-scheduling as follows by a system of parallel independent queues: Let $\gamma_{i,l}$ be the WFQ-weight that determines the proportion of total bandwidth that is given to class i. Naturally, on link l the sum of the WFQ-weights should equal one:

$$\sum_i \gamma_{i,l} = 1 \text{ for all } l.$$

The lower bound of the capacity $b_{i,l}$ of class i on link l depends on the WFQ weights:

$$b_{i,l} = \gamma_{i,l} b_l, \quad \text{for all } i, l, \tag{3.20}$$

resulting in the following packet delays per class:

$$E[D_i] = \frac{1}{\Lambda_l} \sum_l \frac{y_{i,l}}{\gamma_{i,l} b_l - y_{i,l}}. \tag{3.21}$$

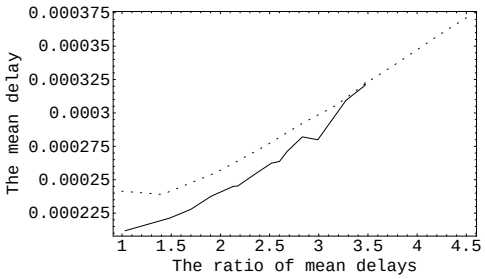

Figure 3.6: The total mean delay as a function of the ratio of mean delays.

Recall the capacity constraint introduced in Section 3.2:

$$y_l \leq b_l, \quad \text{for all } l \in \mathcal{L}.$$

Now this constraint will be replaced by a class-wise capacity constraint

$$y_{i,l} < \gamma_{i,l} b_l, \quad \text{for all } i, l.$$

Also the flow conservation constraint is class-wise:

$$Ax_{i,l} = R_{i,l}, \quad \text{for all } l, (i,j).$$

In our load balancing problem with differentiated service the goal is to find such flow allocations $x_{i,k,l}$ and WFQ-weights $\gamma_{i,l}$ that the sum of weighted mean delays (3.6), where the delays are defined by formula (3.21), is minimized.

To solve the non-linear optimization problem presented above is quite time-consuming. For this reason we introduce a near optimal approach that makes the size of the problem smaller. The algorithm is as follows:

1. Allocate the traffic into the network without WFQ-weights so that the weighted sum of mean delays is minimized as presented in section 3.6.

2. Fix the traffic allocation obtained in the first step.

3. Determine WFQ-weights by minimizing (3.21) with fixed link flows. Now the number of free variables equals the number of WFQ-weights, i.e., the number of links in the network multiplied by the number of classes minus one.

Figure 3.6 depicts a numerical example of the routing that utilizes WFQ-scheduling. In the figure, the solid line represents the result of a straightforward optimization of routing and WFQ-scheduling and the dotted line gives the result of the two-step approach presented above. The straightforward optimization seems to have smaller mean delay. The difference is more significant when the ratio of mean delays is small, but when the ratio is greater, the performance of the two-step algorithm is near to the optimum.

3.7 Load balancing in wireless mesh networks

In the previous sections we studied wired networks in which the fixed links have some predefined capacities and the traffic flows using these links share this capacity. In wireless mesh networks the concept of the link differs from the wired networks. The existence and the capacity of the link, or say virtual link, depends on many factors, such as the distance between the communicating nodes and interference caused by the other nodes sending at the same time.

The idea of load balancing in WMNs is similar to that in fixed networks. Traffic is allocated to the network in such a way that some performance criterion is fulfilled. In addition to taking advantage of spatial load balancing by moving traffic from loaded links to other routes, congestion can temporally be reduced in WMNs by scheduling longer time shares for the loaded links. Optimization in these two dimensions has to be done jointly to achieve the best results.

Wireless network model

We assume that the MAC layer of the wireless network is modelled by Spatial TDMA [NK85]. In STDMA the transmission resources are divided into time slots and the links that are spatially sufficiently separated can transmit in the same slot. Consider a wireless mesh network, which consists of N nodes equipped with a single radio. The distance between nodes i and j is denoted by d_{ij}. The communication range of node i is denoted by R_i and the interference range by R'_i.

Adapting the simple interference model from [JPPQ05], the transmission from node i to j is successful if the following constraints are true:

1. $d_{ij} \leq R_i$.
2. No node k with $d_{kj} \leq R'_k$ is transmitting.

From the above constraints we can form the links of the network as well as the so called *transmission modes*, each mode consisting of a set of links that can transmit at the same time. The mode is said to be *maximal*, if we cannot insert any link to the mode without violating the aforementioned rules for successful transmission. Let $\mathcal{M}$ denote the set of maximal transmission modes and M the number of those modes. In addition, let $S_{l,m}$ denote the nominal capacity of link l in mode m. We assume that the capacities of the active links remain fixed over the transmission modes.

In general, finding all feasible transmission modes is known to be a hard problem. For example, in a network with 22 links, the feasible transmission modes have to be selected from over 4 million different link combinations. However, the maximal transmission modes can effectively be generated by a simple algorithm which adds non-interfering links to a tree structure in a depth-first search manner. The proposal for the algorithm can be found in Publication 7.

In STDMA, to avoid interference, different transmission modes are scheduled in time. A schedule defines the proportion of time that each mode is used for transmission. Let t_m denote the proportion of transmission time of mode m. The sum of the proportions of transmission times of

all modes should equal one:

$$\sum_{m \in \mathcal{M}} t_m = 1.$$

The actual capacities of link l, denoted by c_l, can be determined by weighting the nominal capacity of a link by the length of the mode's time slot in the schedule and taking the sum over all M modes:

$$c_l = \sum_{m \in \mathcal{M}} S_{l,m} t_m.$$

When scheduling is executed on sufficiently fast time scale, the flows using the links in effect experience constant link capacities given above.

Load balancing problem in WMNs

In this subsection we study the static load balancing problem for wireless mesh networks with STDMA scheduling. As compared to the standard load balancing algorithm in Section 3.2, in the wireless context the new thing is that, in addition to congestion management by routing, resources can managed by shifting them from one link to another with scheduling.

In the wireless context, taking the interference described in the previous subsection into account, the capacity constraint introduced Section 3.2 will be replaced by the following constraint:

$$y_l \leq c_l = \sum_{m \in \mathcal{M}} S_{l,m} t_m, \quad \text{for all } l \in \mathcal{L}.$$

The objective in load balancing is then to minimize objective function $C(x)$ by choosing an optimal traffic allocation $x^* = (x_l^k; k \in \mathcal{K}, l \in \mathcal{L})$ and transmission schedule $t^* = (t_m; m \in \mathcal{M})$. If our objective is to minimize the maximum link utilization α, the capacity constraint is (compare to problem (3.7)):

$$y_l \leq \alpha \sum_{m \in \mathcal{M}} S_{l,m} t_m, \quad \text{for all } l \in \mathcal{L},$$

where both α and t_m are free variables. To get the right-hand-side of the constraint linear, we introduce a new variable $q_m = \alpha\, t_m$. The capacity constraint is then:

$$y_l \leq \sum_{m \in \mathcal{M}} S_{l,m} q_m, \quad \text{for all } l \in \mathcal{L}.$$

Now the objective in load balancing is to choose an optimal traffic allocation $x^* = (x_l^k; k \in \mathcal{K}, l \in \mathcal{L}))$ and scaled transmission schedule $q^* = (q_m; m \in \mathcal{M})$ satisfying

$$\sum_{m \in \mathcal{M}} q_m = \alpha.$$

The rest of the problem is similar to the standard load balancing problem presented in Section 3.2. As an example, if our objective is to minimize the maximum link utilization and all possible paths are available, the resulting optimization problem reads:

$$
\begin{aligned}
&\text{Minimize } \alpha \\
&\text{subject to the constraints} \\
&x_l^k \geq 0, && \text{for all } k \in \mathcal{K} \text{ and } l \in \mathcal{L}, \\
&q_m \geq 0, && \text{for all } m \in \mathcal{M}, \\
&\textstyle\sum_{m \in \mathcal{M}} q_m = \alpha, \\
&\sum_{k \in \mathcal{K}} x_l^k \leq \sum_m S_{l,m} q_m, && \text{for all } l \in \mathcal{L}, \\
&Ax^k = R^k, && \text{for all } k \in \mathcal{K}.
\end{aligned}
\tag{3.22}
$$

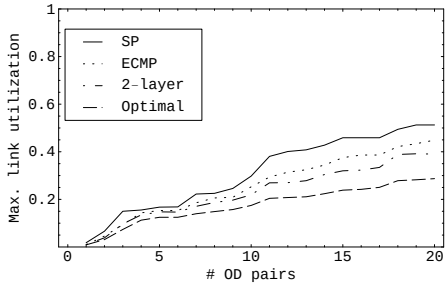

Figure 3.7: The maximum link utilization as a function of the number of OD pair demands. SP, ECMP, 2-layer optimization and optimal.

A numerical example of the load balancing in WMNs is shown in Figure 3.7. The test-network is a 4x4 grid network with 16 nodes, 48 links and 2934 maximal transmission modes. We load the network by random IE pair demands, the number of which varies from one to 20. The cross-layer optimization of the traffic allocation and the schedule is compared to single shortest path routing, ECMP and 2-layer approach in which traffic is first balanced without knowledge of the interference and after that the schedule is optimized. The results show that when the joint optimization is compared with other approaches, the decrease in the maximum load is many times over 50 %. Optimization in the two dimensions, routing and scheduling, has to be done jointly to achieve the best results.

3.8 Summary and Conclusions

In this chapter we have studied load balancing in different types of networks, namely MPLS, OSPF, differentiated services, and WMN networks. The study was started by formulating the well-known static load balancing problem and defining different optimization objectives, as well as describing solution methods.

The main contribution of this chapter was presented in Section 3.5, where an adaptive load balancing algorithm to solve the static load balancing problem is proposed. The algorithm makes incremental changes in the

load distribution on the basis of the measured link loads. Measuring the link counts by SNMP, for example, is much easier than obtaining information about the end-to-end traffic demands.

The general version of the adaptive load balancing algorithm is applied to MPLS and OSPF networks. The difference between these networks is that in MPLS networks each ingress node of the demand defines how traffic is split into parallel paths, whereas in OSPF networks each router makes load balancing decisions along the shortest path from the ingress to the egress node. Decisions are made for the aggregate traffic flow instead of IE pair demands.

The numerical results from experiments with the adaptive algorithm show that in both types of networks the optimal traffic allocation can be achieved. In an MPLS network the algorithm converges to the optimal solution of the static load balancing problem with an arbitrary path set if the number of parallel LSPs is sufficient, whereas in an OSPF network the algorithm converges to the optimal solution of the restricted optimization problem in which only the shortest paths are allowed. Finding the global optimum in an OSPF network requires the link weights to be changed, which is not desirable on the short time scale of traffic engineering where our adaptive algorithm operates.

We also saw that moderate random traffic fluctuations in the measured link loads are tolerable for the convergence of the adaptive algorithm. When the step size $1/g$ is small enough, the algorithm does not react too aggressively to random changes in the link loads. The small step size guarantees conservative changes in control parameters and thus the usage of additional smoothing or filtering is not necessarily required.

In addition to the adaptive algorithm, we also modified the static load balancing problem for achieving service differentiation. Since the differentiation obtained by using the routing only is limited and can lead to poor overall performance, we studied more effective differentiation by sharing the capacities of the links among different classes according to WFQ-weights. In our work, the capacity provided to each class should be considered as lower bound.

Finally, we considered load balancing in wireless mesh networks. In those networks, in addition to spatial load balancing, the resources are shared temporally among different transmission modes. Our approach was to calculate the transmission modes beforehand and after that formulate the load balancing as a rather simple LP problem. A drawback of this approach is the huge number of transmission modes when the number of links in the network increases. However, these modes have to be calculated only once for each topology.

4 PERFORMANCE EVALUATION OF P2P FILE SHARING SYSTEMS

4.1 Introduction

Peer-to-peer (P2P) networks have become a significant area of Internet communication in recent years. The most popular application of those networks is file sharing, in which peers download and upload content with other peers. Older examples of file sharing applications are Gnutella, Napster, and Kazaa, while BitTorrent and its variants have become the most used application in recent years.

Detailed measurement studies of BitTorrent [IUKB+04], [PGES05], have shown that the process of sharing a single file is not stationary but can be divided into different phases. In the first *flash crowd* phase the demand for the newly released file is high. After a couple of days the demand decreases and the flash crowd is followed by a *steady state* in which the demand for the file and the service capacity of the system are in balance. Because of the decentralized character of BitTorrent, there are no guarantees that all the chunks of the file are present in the system over time. If some chunk of the file is missing, the file is not complete any more. Depending on the application, this might or might not be crucial. This third phase of the system lifetime is called the *end phase*.

One key feature of the novel P2P protocols, such as BitTorrent, is that the file is divided into parts, called chunks. While downloading the other chunks, peers at the same time upload the chunks they already have to other peers as *leechers*. With this approach the service capacity of the P2P file sharing network increases as the number of downloaders increases, and the system is scalable against the flash crowd phase. However, the relation between the number of downloaders and the service capacity of the system leads to a feedback system and complicated dynamics of system performance. Most of the analytical models in literature simplify the system by a deterministic fluid model. However, with the use of this approach many characteristics of the system remain concealed.

Evidently, the popularity of the P2P applications, as well as large file sizes, causes congestion in the links of the IP network, especially in its access part. Thus, from an operator's point of view it is important that the traffic load produced by P2P applications is well allocated to the network. Naturally, the load balancing mechanisms introduced in the previous chapter also manage the traffic and congestion induced by the P2P applications. However, in P2P file sharing networks the traffic patterns are not fixed in the traditional manner, since there can be multiple sources for the desired data. Thus, in addition to the network-level load balancing mechanisms, the congestion of the links should be proactively avoided by intelligent peer selection strategies. Depending on the application, congestion management can be initiated by individual peers or a central management system.

In this chapter we study the performance of a P2P file sharing network with analytical models and simulations. First, we concentrate on the evolution of the system when the peers are downloading and uploading a single

chunk. A deterministic fluid model is constructed to capture the overall behavior of the population dynamics of the downloaders and seeds and after that a more detailed Markov model is presented allowing us to estimate the lifetime of the file sharing system, that is, the mean time from the release of a file to its disappearance. Both constant and exponentially decreasing peer arrival rates are studied. Second, we extend the Markov chain model for the case of sharing two chunks and compare different peer selection policies with the aid of this model. Finally, by providing the downloaders and seeds with location information we study further the effect the selection of the peer has on the resource usage of the network.

4.2 Related research

In this section we first review some earlier research aiming to model P2P file sharing systems and evaluating the performance of them. We focus on the BitTorrent-like P2P file sharing systems. Second, literature related to some specific performance optimization problems in other types of P2P networks is presented.

Modelling of BitTorrent-like file sharing systems

Yang and de Veciana study the service capacity of P2P networks in [YdV04]. The performance analysis of the system is divided into two time regimes, transient regime and steady state regime, which are analyzed separately. In the transient regime the growth of the service capacity is exponential and the system is modelled by an age-dependent branching process. The steady state of the system is analyzed by a Markov chain model, where the peer that has the complete file can upload it to other peers with service rate μ and the downloader that has a part of the file can upload it with rate $\eta\mu$. By numerical computations with the Markov model, the paper [YdV04] finds that service capacity increases linearly in the offered load.

Qiu and Srikant [QS04] formulate a deterministic fluid model corresponding to the model of [YdV04]. In addition to parameters of [YdV04], the download process of each peer is constrained by rate c and a peer may abort downloading with rate θ. By the steady state analysis and Little's law, the average downloading time is obtained. The efficiency of uploading, parameter η, is evaluated by assuming a uniform distribution for the number of chunks the peers are having.

The studies of papers [QS04] and [YdV04] do not tell how long the download process of a file actually takes from the user's viewpoint. The analysis of total delay is done in [KR05], where it is divided into three components; queueing delay at core routers, delay produced by peers to acquire service and link propagation delay. To calculate the queueing delay in the routers, a single class open queueing network is formed, where each router is modelled as a GI/G/1 queue. The delays induced by the uploading peers are characterized by modelling each individual peer as an M/G/1/m-PS queue, where m is the number of simultaneous downloaders.

Among others, Fan et al. [FCL06] notice that the model of paper [QS04] hides many details of the file sharing process by assuming, e.g., constant throughput for the peers. Thus in [FCL06] the leechers are divided into

two types; leechers that have a few chunks and leechers that have almost all the chunks. A deterministic fluid model is then formulated to describe the evolution of these two types of leechers as well as seeds. Massoulié and Vojnović [MV05] divide the leechers further into layers such that peers having the same number of chunks are located at the same layer. In the paper, stability conditions of the system, in which the chunks are swapped only between the peers at the same layer, are derived.

Even though the aforementioned models are quite complicated, they do not still capture all characteristics of the P2P sharing process. The arrival rate is assumed to be constant, and especially, all chunks of the file are assumed to be available forever. To the best of our knowledge, the only studies that do not make these assumption are provided in [TWN06] and [GCX+05]. Tian et al. study the integrity of the file in [TWN06]. First, using Markov chain modelling, the steady state solution for the number of downloaders holding a certain number of chunks is derived and then the file availability is estimated by a simple probabilistic model. Paper [GCX+05] first proposes that the arrival rate of file requests decreases exponentially in time and studies then the system by a deterministic fluid model. The lifetime of the system, i.e., the mean time from the appearance of the file until its disappearance is derived by a simple deduction.

Optimization of P2P networks

A lot of research concerning P2P networks concentrates on decentralization of the control by distributing specific items among the peers. Distributed hash tables (DHTs) is one approach to handle the allocation of the responsibilities in a P2P network. One example of those protocols is Chord [SMLNDK03]. Indeed, load balancing in P2P networks often refers to an even sharing of keys among the peers instead of congestion reduction in the links of the underlying network.

An interesting question is how an overlay network formed by a P2P application matches with the physical IP network topology. Ratmasany et al. [RHKS02] point out that a neighboring node in the overlay topology can be located even in a different continent and thus messages traverse along very long routes. To solve this problem, the paper proposes a distributed binding scheme to optimize the overlay topology. In the scheme each node of the network measures the round-trip time to a well-known landmark machine and then selects a peer based on the measurement.

Ng et al. [NhCR+03] investigate measurement-based optimization of P2P networks further. The idea is to use light-weight measurement infrastructure to quantify a good peer for bandwidth-demanding file sharing. In the proposal a peer with highest TCP throughput is selected as a neighbor.

One example of attempts to improve the overlay network of an P2P application is decentralized object location and routing system named Pastry [RD01]. Pastry routing tables forward messages to nodes that are relatively close according to a chosen proximity metric, e.g., the number of IP routing hops.

A basic problem in optimization of P2P networks is the lack of incentives of peers to co-operate. Also elimination of free-riders without a central control is a problem. The study in [FLSC04] tries to tackle these problems

by a game-theoretic approach. In the approach every peer keeps track of the interactions between the other peers and the history is shared between peers to obtain better performance of the system.

4.3 Contribution

In this thesis we provide new analytical models for studying the population dynamics and performance metrics in a P2P file sharing system. In particular, we are focusing on the lifetime of a file sharing system with a constant or fading peer arrival rate. In addition, we provide a model for sharing two chunks which allows us to evaluate different peer selection policies.

In more detail, Publication 8 studies sharing of a single chunk in a P2P file sharing system. A Markov chain model to capture the dynamics of the number of downloaders and seeds is proposed allowing us to estimate the lifetime of the file sharing process. In addition, the underlying topology of the network is modelled by a sphere on which the peers are located. With the resulting spatio-temporal model we assess how much the resource usage of the network can be reduced, e.g., by selecting the nearest seed for download instead of a random one.

Demand for a file hardly remains constant over time in P2P networks, but decreases after the flash crowd phase. In order to account for this fact, in Publication 9 we continue the evaluation of the lifetime of the file sharing process but, instead of a constant peer arrival rate, we assume now that the peer arrival rate decreases exponentially. A time-dependent Markov chain model for the system is developed and exact values for the mean lifetime of the system as well as approximations are given.

Finally, Publication 10 studies how the division of the file into chunks as well as the used chunk selection policy influence the availability and mean download time of the file. For that purpose, a detailed Markov chain model for sharing two chunks is proposed. In addition, we use simulations to evaluate more complex systems with multiple chunks.

4.4 System description

In this section we describe in general terms the system that we are studying. Due to the rapid progression of the P2P file sharing protocols and applications, we do not want to restrict ourselves to specific technical details of any certain protocol, but describe the general idea of those systems. For keeping the system simple, some details of the protocols, such as the choking algorithm in BitTorrent, are excluded from the discussion.

Let L be the size of the file under consideration in bytes. The file is divided into blocks, called chunks, which are assumed to be downloaded one after another separately. If we fix the size of a chunk to l bytes, the total number of chunks, denoted by K, is naturally $\lceil L/l \rceil$.

In the system we have three types of peers, *downloaders*, *leechers*, and *seeds*. Downloaders do not have any chunk yet and try to download a first one. Leechers have already downloaded a part of the chunks and can upload those chunks to other peers in the system. While uploading the chunks, they try to find the missing chunks. Finally, the peers that have all

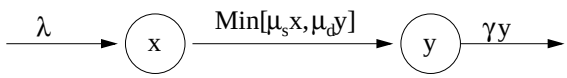

Figure 4.1: A model for sharing a chunk.

chunks are called *seeds*.

When a new downloader has arrived, it seeks a peer and a chunk among all available peers (including leechers and seeds) and chunks according to a given chunk selection policy. Let μ_d denote the download rate and μ_s the upload rate of a peer. We fix the mean download time of a chunk to be proportional to the chunk size, that is, $1/K\mu_d$. Also the mean upload time of a chunk is proportional to the chunk size, $1/K\mu_s$. If there are many downloaders per one leecher or seed concurrently, we assume that the upload capacity of the peer is shared among the downloaders. We also assume that the time required for finding the peer for exchanging the chunks is negligible as compared to the download time. If the leecher or seed of the downloader leaves before the download process for a chunk is completed, the downloader has to find a new peer and start to download the chunk again from the beginning.

After the peer has downloaded the first chunk, it seeks the second chunk and starts to download that. It can also simultaneously upload the chunk it has, to other peers as a leecher if required. When the leecher has collected all K chunks, the status of the peer changes from a leecher to a seed. We assume that the seeds stay in the system for a random, exponentially distributed period with parameter γ (the departure rate of a seed).

4.5 A single chunk model for P2P file sharing

In this section we analyze the population dynamics of the file sharing system described in the previous section by concentrating on a single chunk of the file. We assume that the disappearance of the chunk means the end of the whole file sharing process. The work is motivated by the model of [QS04] but differs from it in some aspects. Paper [QS04] assumes that peers can upload the chunks with a constant rate independent irrespective of the requests for them. However, we find the assumption unrealistic and the model probably hides some details of the population dynamics. For this reason we consider the sharing of a single chunk at a time. In addition, among others, papers [YdV04] and [QS04] assume that at least one seed stays in the system keeping the chunks available. This assumption is not necessarily true in distributed systems without central control.

Constant peer arrival rate

In this subsection new requests for a chunk are assumed to arrive at the system with rate λ according to the Poisson process. The downloader can download the chunk with rate μ_d. On the other hand, the maximum upload rate of a peer for the chunk is assumed to be μ_s. After the download, the status of the downloading peer changes from a *downloader* directly to a

seed and the peer can distribute the chunk further. Note that in this context, a peer is referred to as the seed if it has the chunk in question, but not necessarily all the chunks of the file. The seed leaves the system with probability γ per time unit. See Figure 4.1 for illustration of the model.

Let $x(t)$ be the number of downloaders and $y(t)$ be the number of seeds at time t in the single chunk model. When $\mu_d x(t) < \mu_s y(t)$, the downloaders cannot use all the service capacity provided by the peers. On the other hand, when $\mu_d x(t) > \mu_s y(t)$ the upload capacity of the seeds limits the download process. Thus the total service rate of the system is $\min\{\mu_d x(t), \mu_s y(t)\}$. A deterministic fluid model for the number of downloaders $x(t)$ and seeds $y(t)$ is then:

$$\begin{aligned} \frac{dx(t)}{dt} &= \lambda - \min\{\mu_d x(t), \mu_s y(t)\}, \\ \frac{dy(t)}{dt} &= \min\{\mu_d x(t), \mu_s y(t)\} - \gamma y(t), \end{aligned} \tag{4.1}$$

with initial conditions $y(0) = 1$ and $x(0) = 0$. Let $\bar{x}$ and $\bar{y}$ be possible equilibrium values of $x(t)$ and $y(t)$. If $\bar{x} \leq \bar{y}$, the steady state solution is $\bar{x} = \lambda/\mu_d$ and $\bar{y} = \lambda/\gamma$. From the constraint $\bar{x} \leq \bar{y}$ we obtain the condition for the equilibrium: $\mu_s \geq \gamma$. When $\mu_s < \gamma$ the solution of the equations (4.1) is $\bar{y} = 0$ and $\bar{x} \to \infty$.

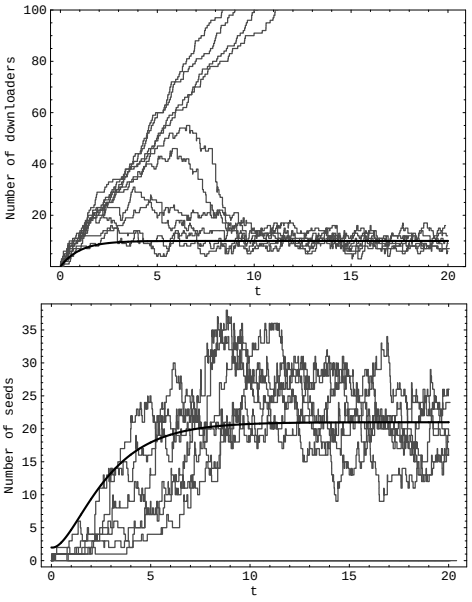

Figure 4.2: The number of downloaders in the upper graph and the number of the seeds in the lower graph as a function of time (in units of $1/\mu_d$). Solid lines: fluid model (4.1), gray lines: simulation. $\lambda/\mu_d = 10$, $\lambda/\gamma = 20$, $\mu_s \geq \gamma$.

The evolution of the number of downloaders and seeds is depicted in Figure 4.2. We have fixed λ/μ_d and λ/γ to moderately small values in order to better demonstrate the dynamics of the system. The solid line

corresponds to the solution of fluid model (4.1) and the gray lines to 10 different simulation traces. We can see that in the beginning the capacity of the system is not sufficient to serve the chunk requests. This is seen as a dramatic increase in the number of downloaders. However, after the status of some downloaders has changed to a seed, the system stabilizes. At the end time ($t = 20$) 4 simulated processes of 10 have become extinct and the number of downloaders in those processes increases without any limit.

The above deterministic fluid model describes the average behavior of the sharing of the chunks and does not capture the dynamic nature of the system in details. For example, if the original seed can leave the system, the death of the chunk and the whole file sharing process is unavoidable, even if $\mu_s > \gamma$. For this reason we study the evolution of the process (x, y) in more detail by a Markov chain model with absorbtion. We construct a continuous time Markov chain process, where the state is the pair (x, y) and the transition rate matrix is Q with the elements:

$$\begin{array}{lll} q((x,y),(x+1,y)) & = \lambda, & \\ q((x,y),(x-1,y+1)) & = \min\{\mu_d x, \mu_s y\}, & \text{if } x > 0, \\ q((x,y),(x,y-1)) & = \gamma y, & \text{if } y > 0. \end{array}$$

All states (x, y) with $y = 0$ in the Markov chain are absorbing and can be combined into one absorbing state, state 0. The mean time to absorbtion is determined as follows: Let b_i denote the mean time to absorbtion, when the system starts from state $i = (x, y)$. Given the transition matrix Q, the mean times to absorbtion b_i are determined by a familiar Markovian recursion:

$$b_i = \frac{1}{q(i)}(1 + \sum_{j \neq i} q(i,j) b_j), \tag{4.2}$$

where mean time to absorbtion from the absorbing state itself, b_0, is zero, and $q(i) = \sum_{i \neq j} q(i, j)$. The absorbtion time starting from the initial state $(0, 1)$, i.e. the lifetime of the system, as a function of λ/γ is shown in Figure 4.3. The solid line is calculated by solving the set of linear equations (4.2) numerically in a truncated state space of 35×35 states. The dots are obtained from simulation of the corresponding infinite system verifying the analytical results. The figure shows that the system lifetime increases exponentially as a function of the expected number of the seeds λ/γ in the system.

In a limiting case the absorbtion time can easily be calculated in closed form. When the mean service times $1/\mu_s$ and $1/\mu_d$ are very small, the system can be modelled as an M/M/∞-queue with arrival rate λ and departure rate γ. The mean time to absorbtion equals the length of the busy period $E[B]$ of the M/M/∞-queue:

$$E[B] = \frac{1}{\lambda}(e^{\lambda/\gamma} - 1).$$

Above Markov model describes the transient behavior of the P2P file sharing allowing us to estimate the mean lifetime of the system. However, when some other performance metrics are considered, such as the total download time, a model with the steady state distribution is better. The

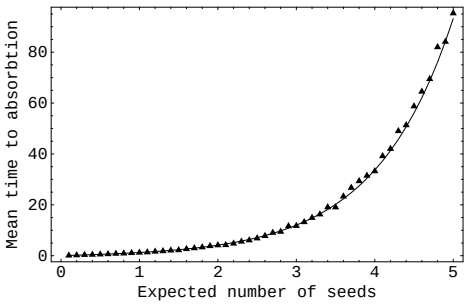

Figure 4.3: The mean time to absorbtion (in units of $1/\lambda$) as a function of the expected number of seeds, λ/γ. Solid line: analytical results, dots: simulation.

earlier Markov model can be easily be modified to cover the steady-state behavior of the system by assuming that there is always at least one seed in the system keeping the chunk available. In addition, if we assume that service is always constrained by the download rate, the process reduces to a more simple model, where the system of downloaders and seeds can be considered as two consecutive queues. Arrival rates to the first and second queues are λ and the departure rates are $x\mu$ and $y\gamma$, respectively. Note that now y excludes the original seed. The system is thus a non-absorbing M/M/∞-tandem queue. The equilibrium distribution of the system is:

$$p\{X = x, Y = y\} = \frac{(\frac{\lambda}{\mu})^x}{x!} e^{-\lambda/\mu} \frac{(\frac{\lambda}{\gamma})^y}{y!} e^{-\lambda/\gamma}. \tag{4.3}$$

Decreasing peer arrival rate

In the studies such as [YdV04] and [QS04] the inter-arrival time of new peers is assumed to be exponentially distributed with a constant arrival rate. However, from measurement studies it is evident that the peer arrival rate decreases over time. For that reason we study next the population dynamics of a P2P system with time-dependent arrival rate $\lambda(t)$ of requests of the chunk. Paper [GCX+05] proposes a traffic model for P2P networks, where the arrival rate decreases exponentially starting from the release of the file:

$$\lambda(t) = \lambda_0 e^{-t/\tau},$$

where parameter τ describes the attenuation of the demand over time. The average number of all requests expected to arrive in the system, denoted by N, is then:

$$N = \int_0^\infty \lambda_0 e^{-t/\tau} dt = \lambda_0 \tau.$$

Similarly to the previous subsection, the evolution of the number of downloaders and seeds, the pair (x, y), can be described by a deterministic fluid model:

$$\begin{aligned}\frac{dx(t)}{dt} &= \lambda_0 e^{-t/\tau} - \min\{\mu_d x(t), \mu_s y(t)\},\\ \frac{dy(t)}{dt} &= \min\{\mu_d x(t), \mu_s y(t)\} - \gamma y(t),\end{aligned} \tag{4.4}$$

where $y(0) = 1$ and $x(0) = 0$ meaning that at the beginning, $t = 0$, there is one seed and no downloaders. Assuming that the mean download and upload times are the same, $\mu_s = \mu_d = \mu$, we find three characteristic solutions for the system:

1. $\mu < \gamma$. The existing seeds leave the system faster than new seeds arise resulting in the steady-state solution $\bar{x} \approx N$ and $\bar{y} = 0$ (see top of Figure 4.4). Note that these N downloaders do not receive the chunk at all.

2. $\mu = \gamma$. The number of seeds stays constant and the number of downloaders increases until the arrival rate attenuates and all peers are served (see the middle of Figure 4.4). The steady-state solution is thus $\bar{x} = 0$ and $\bar{y} = 0$.

3. $\mu > \gamma$. The number of both the downloaders and the seeds increases until all downloaders are served resulting in $\bar{x} = 0$ and $\bar{y} = 0$ (see the bottom of Figure 4.4).

The time-dependent arrival rate changes also the Markovian analysis of the system. Let $\pi_{(x,y)}(t)$ denote the probability of state (x, y) at time t. The time-dependent Kolmogorov's forward equations for transition rates between the states are the following:

$$\begin{aligned}
\tfrac{d}{dt}\pi_{(x,y)}(t) &= \lambda(t)\pi_{(x-1,y)}(t) + \min\{\mu_d x + 1, \mu_s y - 1\}\pi_{(x+1,y-1)}(t)\\
&\quad +(y+1)\gamma\pi_{(x,y+1)}(t)\\
&\quad -(\lambda(t) + \min\{\mu_d x, \mu_s y\} + \gamma y)\pi_{(x,y)}(t), \text{ for } x \geq 1, y \geq 1,\\
\tfrac{d}{dt}\pi_{(x,0)}(t) &= \lambda(t)\pi_{(x-1,0)}(t) + \gamma\pi_{(x,1)}(t) - \lambda(t)\pi_{(x,0)},\\
&\quad \text{for } x \geq 1, y = 0,\\
\tfrac{d}{dt}\pi_{(0,y)}(t) &= \pi_{(\mu_d 1, \mu_s y - 1)}(t) + (y+1)\gamma\pi_{(0,y+1)}(t)\\
&\quad -(\lambda(t) + \gamma y)\pi_{(0,y)}(t), \text{ for } x = 0, y \geq 1,\\
\tfrac{d}{dt}\pi_{(0,0)}(t) &= \gamma\pi_{(0,1)}(t) - \lambda(t)\pi_{(0,0)}(t), \text{ when } x = 0, y = 0,
\end{aligned}$$

with initial condition $\pi_{(x,y)}(0) = 1$, when $x = 0$ and $y = 1$, and $\pi_{(x,y)}(0) = 0$ otherwise. The explicit solutions for the state probabilities $\pi_{x,y}$ are very complicated but the numerical solutions can easily be found using a differential equation solver.

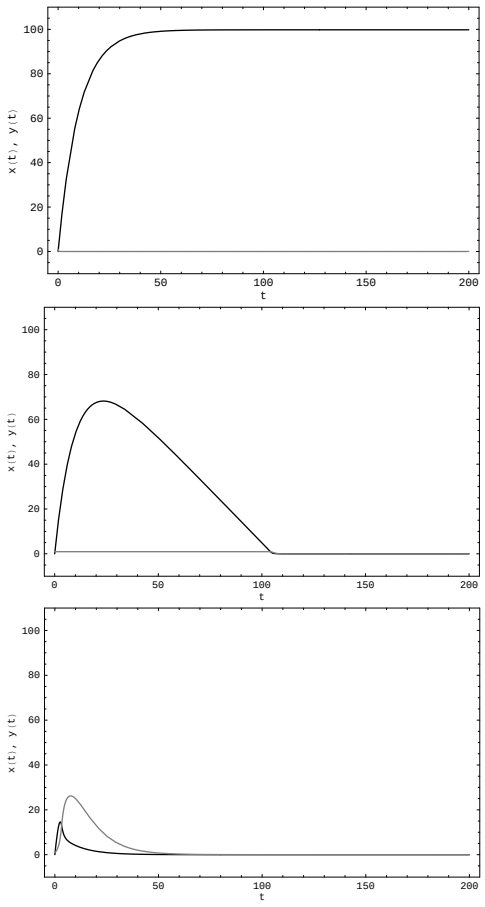

Figure 4.4: The number of downloaders and seeds as a function of time, when $\lambda_0 = 10$, $\tau = 10$, $\mu = 1$ and $\gamma = 5$ (top), $\gamma = 1$ (middle) and $\gamma = 1/5$ (bottom). Black lines: downloaders, gray lines: seeds.

Also in this system the states (x, y) with $y = 0$ are absorbing. Let $z_{(x,y)}$ denote the mean time spent in state (x, y) from the beginning ($t = 0$) to the absorbtion of the system:

$$z_{(x,y)} = \int_0^\infty \pi_{(x,y)}(t)dt.$$

The mean time to absorbtion, i.e. the lifetime of the system, is thus the sum of the time spent in the non-absorbing states:

$$T_{life} = \sum_{(x,y):y>0} z_{(x,y)}. \tag{4.5}$$

In the upper graph of Figure 4.5 the mean lifetime of the chunk sharing process is shown as a function of $1/\gamma$ for relatively small values of $1/\gamma$. In the figure $\lambda_0 = 10$ and $\mu_d = \mu_s = 1$ and τ varies from 10 to ∞. We can see

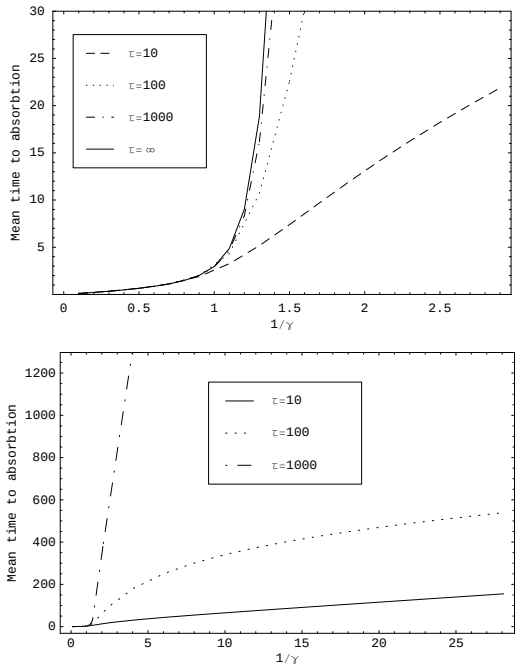

Figure 4.5: The mean lifetime of the chunk sharing process when $1/\gamma$ varies from 0 to 3 (upper graph) and from 0 to 30 (lower graph). The attenuation parameter τ varies from $\tau = 10$ to $\tau \to \infty$. Parameters $\lambda_0 = 10$ and $\mu_s = \mu_d = 1$.

that the mean lifetime of the system increases exponentially as a function of the mean time the seeds stay in the system, $1/\gamma$. For example, when $\tau = 1000$, the mean lifetime of the system is close to that of the system with constant arrival rate (case $\tau = \infty$). However, when $1/\gamma$ increases (see the lower graph of the same figure), the growth of the mean lifetime as a function of $1/\gamma$ is only linear when $\tau < \infty$.

Also in the case of decreasing arrival rate, the lifetime can be approximated in a limiting regime, where the mean time that the seeds spend in the system is long as compared to the length of the burst of the arrivals. First, if we wait proportion p of total N peers expected to arrive, the length of the burst period, denoted by T_1, can be calculated to be:

$$T_1 = \tau \log(\frac{1}{1-p}).$$

Second, the mean departure time of all N seeds is

$$T_2 = \frac{1}{N\gamma} + \frac{1}{(N-1)\gamma} + \frac{1}{(N-2)\gamma} ... + \frac{1}{\gamma} = \sum_{i=0}^{\lfloor N \rfloor} \frac{1}{i\gamma}.$$

Assuming that no seed has left the system before the end of the arrival burst, the lifetime of the chunk sharing system can be approximated by the

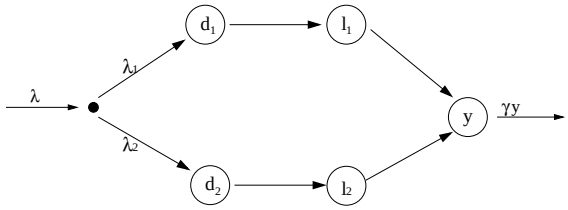

Figure 4.6: A model for two chunks.

sum of these two consecutive time periods, T_1 and T_2:

$$T_{life} \approx T_1 + T_2.$$

4.6 Markov model for two chunks

A detailed model of the system with multiple chunks is very complicated and hard to solve. On the other hand, more simple models, such as deterministic fluid models, describe only the average behavior of the sharing of the chunks and are thus unable to capture all the details of the process such as possible unstability and extinction of the system. For this reason, in this section we propose a detailed Markov chain model for the P2P file sharing process when there are two chunks. Even with this limited model we are able to analyze different chunk selection policies. In particular, in this thesis we compare:

- *Random peer selection* (denoted by RND peer). First the peer for downloading is randomly selected among all peers in the system and after that a random chunk from the selected peer's chunk collection is downloaded.
- *Random chunk policy* (denoted by RND chunk). First the chunk to be downloaded is selected randomly among all chunks and then a random peer having this chunk is selected.
- *Rarest chunk first policy* (denoted by RF). The chunk that has least replicas in the system is first selected for downloading. The peer to upload this chunk is selected randomly among all peers that have the chunk. RF chunk selection is used in BitTorrent, for example.
- *Most common chunk first* (denoted by MCF). The chunk that is most common is selected for download. The peer to upload this chunk is selected randomly among all peers that have the chunk.

The chunks of the file are named chunk 1 and chunk 2. Correspondingly, we have two types of downloaders in the system; downloaders that are downloading chunk 1 and donwloaders that are downloading chunk 2. Also the leechers can be divided into two groups; leechers that have chunk 1 and leechers that have chunk 2. Let $d_1(t)$ denote the number of downloaders for chunk 1 and $d_2(t)$ downloaders for chunk 2 at time t. These peers do not have any chunk. Let $l_1(t)$ and $l_2(t)$ denote the number of

leechers of type 1 or 2, respectively. Finally, as in the previous section, let $y(t)$ be the number of seeds having both chunks at time t. Let the quintet $\mathbf{x} = \{d_1, d_2, l_1, l_2, y\}$ be the state of the system. In Figure 4.6 we have illustrated the model by a simple flow diagram.

The download time of a chunk depends both on the load of the peer that is uploading to the downloader and on the downloader's own download capacity. We assume an idealistic model in which the maximum download rate of a peer for a chunk is $2\mu_d$ but the actual received rate is the fair share of the total upload capacity provided by the leechers and seeds. This share depends both on the momentary number of leechers and seeds for the given chunk, and on the number of peers that are currently requesting the chunk. If we consider sharing of upload capacity of chunk 1, $d_1 + l_2$ peers download the chunk and l_1 leechers and y seeds provide it. Since the seeds have both chunks, the capacity of them is divided among all peers downloading something. Finally, the fair share of the upload capacity of leechers and seeds per downloader for chunk 1 is denoted by $\phi_1(\mathbf{x})$ and can be written as

$$\phi_1(\mathbf{x}) = (\frac{l_1}{d_1 + l_2} + \frac{y}{d_1 + d_2 + l_1 + l_2})2\mu_s. \tag{4.6}$$

Similarly, the shared upload capacity for chunk 2 is:

$$\phi_2(\mathbf{x}) = (\frac{l_2}{d_2 + l_1} + \frac{y}{d_1 + d_2 + l_1 + l_2})2\mu_s. \tag{4.7}$$

Note that in a real system the downloader is hardly able to utilize all upload capacity of the leechers and the seeds due to mismatch of the peers. However, this assumption makes the analytical model solvable.

The evolution of the downloaders, different type of leechers and the seeds can be described by a five-dimensional Markov chain, whose state is $\mathbf{x} = (d_1, d_2, l_1, l_2, y)$ and transition rate matrix is Q with the following elements:

$$\begin{aligned}
&q(\mathbf{x}, \mathbf{x} + e_{d_1}) = \lambda_1(\mathbf{x}),\\
&q(\mathbf{x}, \mathbf{x} + e_{d_2}) = \lambda_2(\mathbf{x}),\\
&q(\mathbf{x}, \mathbf{x} - e_{d_1} + e_{l_1}) = \min\{2\mu_d d_1, \phi_1(\mathbf{x})d_1\},\\
&q(\mathbf{x}, \mathbf{x} - e_{d_2} + e_{l_2}) = \min\{2\mu_d d_2, \phi_2(\mathbf{x})d_2\},\\
&q(\mathbf{x}, \mathbf{x} - e_{l_1} + e_y) = \min\{2\mu_d l_1, \phi_2(\mathbf{x})l_1\},\\
&q(\mathbf{x}, \mathbf{x} - e_{l_2} + e_y) = \min\{2\mu_d l_2, \phi_1(\mathbf{x})l_2\},\\
&q(\mathbf{x}, \mathbf{x} - e_y) = \gamma y,
\end{aligned} \tag{4.8}$$

where e_i is a vector the elements of which are zeros, except element i, which is 1. The arrival rate of type 1 and 2 downloaders, denoted by λ_1 and λ_2, respectively, depends both on the chunk selection policy and the state of the system. The arrival rates for each policy are explained next:

- **Random peer selection**

  In the RND peer policy the selected peer is a leecher with chunk 1 with probability $\frac{l_1}{l_1+l_2+y}$, a leecher with chunk 2 with probability $\frac{l_2}{l_1+l_2+y}$ and a seed with probability $\frac{y}{l_1+l_2+y}$. The seeds have both chunks 1 and 2, so if the downloader selects a seed for chunk exchange, the probabilities to get chunk 1 or chunk 2 are equal. Therefore, the total arrival rate for chunk 1 downloaders is

  $$\lambda_1(\mathbf{x}) = \frac{l_1 + \frac{y}{2}}{l_1 + l_2 + y}\lambda,$$

  and for chunk 2 downloaders

  $$\lambda_2(\mathbf{x}) = \frac{l_2 + \frac{y}{2}}{l_1 + l_2 + y}\lambda,$$

  where λ is the arrival rate of new requests of the file.

- **Random chunk selection**

  In the RND chunk policy peers first select a chunk randomly among all available chunks. In the case of two chunks the arrival rates for different type of downloaders are thus $\lambda_1(\mathbf{x}) = \lambda_2(\mathbf{x}) = 0.5\lambda$.

- **Rarest chunk first selection**

  In RF, if $l_1 < l_2$, chunk 1 is rarest and new peers start to download chunk 1. In that case $\lambda_1(\mathbf{x}) = \lambda$ and $\lambda_2(\mathbf{x}) = 0$. On the other hand, if $l_1 > l_2$, all downloaders start to download chunk 2 and $\lambda_1(\mathbf{x}) = 0$ and $\lambda_2(\mathbf{x}) = \lambda$. If $l_1 = l_2$, half of the downloaders download chunk 1 and the other half chunk 2 and thus $\lambda_1(\mathbf{x}) = \lambda_2(\mathbf{x}) = 0.5\lambda$.

- **Most common chunk first**

  MCF is exactly opposite to the RF policy; if $l_1 > l_2$, then $\lambda_1(\mathbf{x}) = \lambda$ and $\lambda_2(\mathbf{x}) = 0$; if $l_1 < l_2$, then $\lambda_1(\mathbf{x}) = 0$ and $\lambda_2(\mathbf{x}) = \lambda$; and if $l_1 = l_2$, then $\lambda_1(\mathbf{x}) = \lambda_2(\mathbf{x}) = 0.5\lambda$.

The file sharing process with two chunks dies if there are no seeds in the system and leechers together do not have the complete file. Thus the states $\mathbf{x} = (d_1, d_1, l_1, l_2, y)$ with $y = 0 \cap (l_1 = 0 \cup l_2 = 0)$ are absorbing. Given the transition matrix Q, the mean time to absorption can be solved by the Markovian recursion (4.2).

Numerical examples of the model are depicted in Figure 4.7. In the upper graph we show the mean absorption times as a function of the average seed departure time $1/\gamma$ for the single chunk model and the two-chunk model with various selection policies, when initially there is one seed ($y(0) = 1$) and no downloaders or leechers. We can see that the division of the file into two pieces improves the file availability since the mean absorption time is longer. As peer selection policies are compared, the performance of the RF policy is slightly better than that of the others. In the lower graph of Figure 4.7 the mean absorption times of the system are depicted as a function of arrival rate λ. The result is that the system with a greater demand has also a longer lifetime since there are more leechers keeping the file available. Also the advantages of using the RF policy are greater as λ increases.

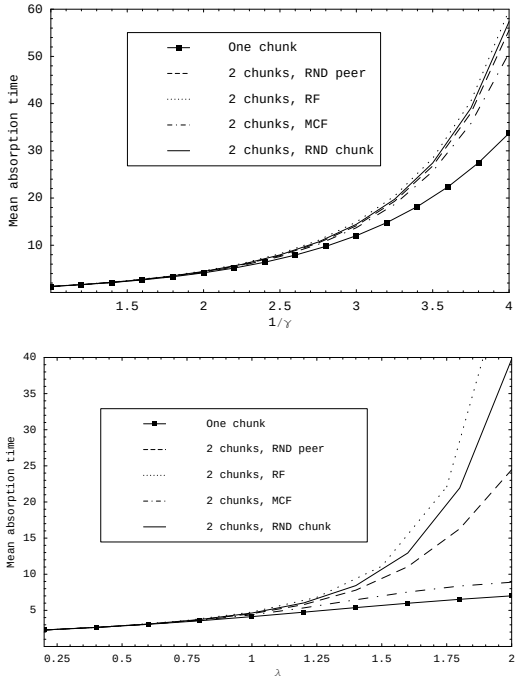

Figure 4.7: The mean absorption time as a function of $1/\gamma$ (the upper graph, $\lambda = 1$) and as a function λ (the lower graph, $\gamma = 1$). Download and upload rates $\mu_d = \mu_s = 1$.

4.7 Simulation of the P2P file sharing system

Earlier in this chapter we considered first sharing of a single chunk as a process independent of the other chunks and then interaction of two chunks. Detailed analytical models for more complicated systems are hardly solvable. One possibility to study more complicated systems with multiple chunks is to use simulations.

Our simulation model corresponds to the detailed description of the file sharing system in Section 4.4. In the simulation we do not assume exponential distribution for download times as in the Markov model but the download time with full download rate is assumed to be deterministic, which reflects better the reality. We study how the division of the file into chunks and the peer selection policy affect the mean lifetime of the system and the mean download time of the file. To achieve reliable and smooth results, simulation process is replicated sufficiently many times. However, for clarity reasons, the confidence intervals are excluded from the figures.

As results of the simulations, the upper graph of Figure 4.8 shows the mean lifetime of the file sharing when the number of chunks, K, varies from 1 to 100. The used chunk selection policy is RND peer. The finding from the simulations is that dividing the file into two pieces improves the lifetime as compared to sharing the file as one chunk. However, there are

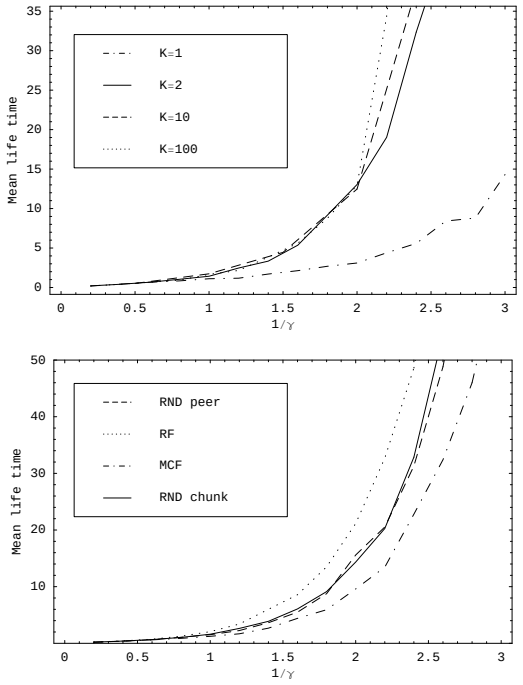

Figure 4.8: The mean lifetime as a function of $1/\gamma$. Upper graph: Random chunk selection, K varies. Lower graph: $K = 10$, the chunk selection policy varies. Download and upload rates $\mu_d = \mu_s = 1$ and $\lambda = 2$.

not so significant differences in the results beyond that; using 10 chunks provides almost as good result as using 100 chunks. Thus also the analytical results for 2 chunks are meaningful with regard to real systems having more chunks. In the lower graph of Figure 4.8 the chunk selection policies are compared. The simulation result validates the result of the analytical model: by using the RF chunk selection the mean lifetime is longest.

Next we study the mean download time of the file. To obtain consistent results, we focus on a non-absorbing system, in which at least one seed is assumed always to be present. In Figure 4.9 the mean download time is depicted as a function of the arrival rate λ, when $K = 2$ (the upper graph) and $K = 10$ (the lower graph). The reference value is $1/\mu_d = 1$, which is the download time with full download rate. The results show that an increase in the arrival rate increases the mean download time only in the beginning. Increasing the number of chunks from 2 to 10 improves the scalability of the system a lot.

As regards to the chunk selection policies, we conclude following: First, when the number of the chunks is small, like $K = 2$, and the arrival rate of the new peer is high, using the RND peer policy or the MCF policy can lead to poor performance. The reason is that in some situations when the number of type-1 leechers is bigger than that of type 2, the probability to select type-1 leecher is bigger and the number of type-1 leechers increases

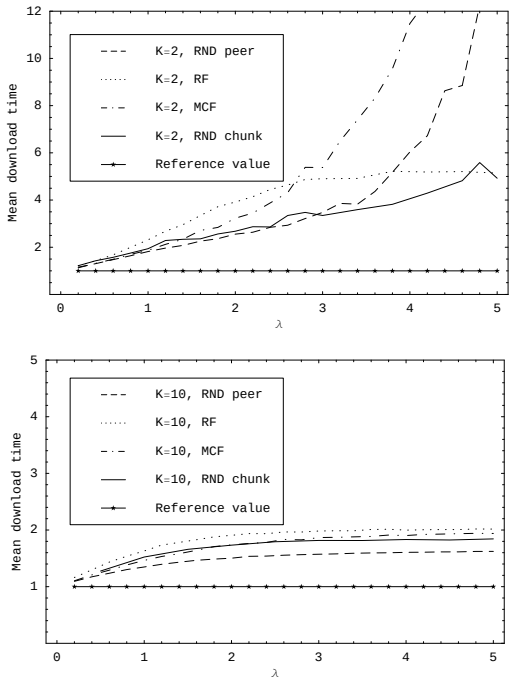

Figure 4.9: The mean download time as a function of λ, when $\gamma = 1/2$ and $\mu_d = \mu_s = 1$. Upper graph: $K = 2$, lower graph: $K = 10$.

until all leechers are of type 1. Then the bottleneck of the system is in downloading chunk 2 from the small number of seeds. If the RF policy is used, there is a balance between type 1 and 2 leechers and performance is better.

However, when the arrival rate is low or we increase the number of chunks to 10, for example, the RND peer and MCF policies lead to better performance than RF. The explanation for this can be found from two things: First, the RND peer policy utilizes the seeds and leechers more evenly. Second, in the RF policy some of the service capacity of the leechers having common chunks probably remains unutilized.

The fourth policy, the RND chunk policy, is a compromise between the other policies. For a small λ, it gives better results than RF and for a big λ it is better than the RND peer or MCF. The benefit of this approach is that one does not have to find out which chunk is rarest but can just select a random one.

4.8 Spatio-temporal modelling of P2P file sharing

Selecting the peer for download randomly or using the RF policy can lead to inefficient use of network resources. A peer may download the chunk from the peer located in an other continent, even if there would be a local neighbor providing the same piece of the file. Long distances in the file

sharing encumber the underlying network unnecessarily much.

In this section we extend the analysis of sharing of a single chunk presented in Section 4.5 by proposing a spatio-temporal model, in which the underlying topology of the network is abstracted by a simple geometrical structure. By this approach we are able to evaluate the location-aware peer selection policies analytically.

As before, in this section new requests for a chunk are assumed to arrive with constant rate λ according to the Poisson process. A new aspect is that each request is associated with a peer i, whose location is assumed to be randomly chosen from the uniform distribution on the surface of a sphere. We have chosen the spherical geometry primarily because it is symmetrical and has no artificial boundaries. It is also a natural choice if one considers a global network. Let R be the radius of the sphere and let the location of peer i be described by cylindrical coordinates z_i and ϕ_i. It is easily verified that if $z_i = -R + 2Ru$ and $\phi_i = 2\pi u'$, where u and u' are drawn from the uniform distribution $U(0,1)$, the peers are uniformly located on the sphere.

Let $D(t)$ be the *set* of downloaders and $S(t)$ be the *set* of seeds at time t. Let parameter p_i denote the selected seed j of downloader i. As a metric for distance between two peers i and j we use the shortest path between the peers on the surface of the sphere, denoted by $d_{i,j}$.

Consumption of the resources of the underlying network is assumed to be proportional to the distance between the peers exchanging chunks. If the peers are far apart, transferring a chunk typically needs more links than in the case of two close peers. Let $c(t)$ denote the total instantaneous capacity required for sharing chunks at time t, $c(t) = \sum_{i \in D(t), j=p_i} d_{i,j}$, i.e., $c(t)$ describes the sum of distances between the peers sharing chunks. When the resource usage optimization is considered, an interesting quantity is the average capacity usage C *per downloaded chunk* over time period $[t_0, t_{max}]$ defined as

$$C = \frac{1}{n} \int_{t_0}^{t_{max}} c(t)dt,$$

where n is the number of chunks transferred within this period.

We consider two different peer selection policies: Random peer selection policy, where the seed for download is selected randomly among all available peers, and the nearest peer selection policy, where the nearest possible peer in terms of the distance between the peers is selected. The expected capacity usages with different policies are derived next.

In the random peer selection policy, the distance to a random seed is independent of the number of seeds. Assuming, without loss of generality, that the mean download time $1/\mu_d$ of a chunk is one, the expected resource usage per a downloaded chunk is equal to the average distance between two points on a sphere (assumed to have unit area): $E[C] = E[d] = \sqrt{\pi}/4$.

In the nearest peer policy, the closest peer among $y(t)$ seeds is selected for download. If N points are randomly distributed on a sphere with unit area, the expected distance to the nearest neighbor can easily be deter-

mined,

$$E[d|N=n] = \frac{\Gamma(n-\frac{1}{2})}{2\Gamma(n)} = \frac{\sqrt{\pi}}{2}\prod_{i=0}^{n-2}\frac{i+\frac{1}{2}}{i+1},$$

which is very accurately approximated by $E[d|N=n] \approx \frac{1}{2\sqrt{n-0.73}}$. At time t, N includes $y(t)$ seeds and the downloader itself, meaning that $N = y(t)+1$. The expected resource usage is then:

$$E[C] = \sum_{y=0}^{\infty} p\{Y=y\}E[d|N=y+1].$$

In general, the steady state distribution of $y(t)$, $p\{Y=y\}$, can be calculated from the Markov model of section 4.5, but the problem is that the solution cannot be expressed in a closed form. However, in the second limiting case introduced in Section 4.5, the steady-state distribution of the downloaders and the seeds excluding the original seed is known. The expected resource usage in this case can be expressed as follows:

$$E[C] = \sum_{y=0}^{\infty} \frac{(\frac{\lambda}{\gamma})^y}{y!}e^{-\frac{\lambda}{\gamma}}\frac{\sqrt{\pi}}{2}\prod_{i=0}^{y}\frac{i+\frac{1}{2}}{i+1}.$$

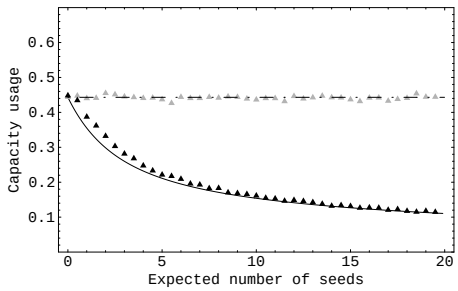

Figure 4.10: Mean capacity usage as a function of λ/γ, $\mu_d = \mu_s = 1$. Gray triangles: simulation of the random peer policy, black triangles: simulation of the nearest peer policy. Dashed line: Expected resource usage for the random peer policy policy and solid line: Expected resource usage for the nearest peer policy policy.

Next we study by numerical examples how the selected policy affects the capacity usage. In Figure 4.10 we study the limiting scenario above, in which the total service rate is always constrained by download rate and at least one peer stays in the system. The capacity usage C is shown as a function of the expected number of seeds λ/γ (simulation starts at time 0, $t_0 = 1000$ and $t_{max} = 10000$). Gray triangles correspond to a simulation with the random peer selection policy and black triangles to the nearest peer selection policy. When λ/γ is small, seeds leave the system shortly after the download and the peers that want to download the chunk have to request it from the original seed. The distances from a downloader to

the original seed using the two different policies are then the same. When λ/γ increases the number of seeds also increases and the selected policy has an effect on the resource usage. We can see that, e.g., for $\lambda/\gamma = 20$ the capacity usage of the policy nearest peer selection is only 23 % of the capacity usage of the random policy. Simulation results are very close to analytical bounds, especially when $\lambda/\gamma > 5$.

4.9 Summary and Conclusions

In this chapter we have studied the population dynamics and performance of novel P2P file sharing networks, in which peers can upload the chunks they already have to other peers. A detailed, state-dependent model of the system with multiple chunks is very complicated and hard to solve. On the other hand, simpler models, such as deterministic fluid models, describe only the average behavior of the sharing of the chunks and are thus unable to capture the dynamic nature of the file sharing process from the appearance of the file until its disappearance. For that reason we studied P2P file sharing by means of two different approaches. In the first one we concentrated on the evolution of the dynamics of sharing a single chunk and in the second one we considered sharing a file as two chunks.

The evolution of the number of downloaders and seeds sharing a single chunk was first studied by the deterministic fluid model. Then, as a novel approach, we evaluated the lifetime of the file sharing process by developing a Markov chain model for a single chunk and solving the mean absorption time of the system. We found that the the system is vulnerable to dying if the seed departure time $1/\gamma$ is short or the peer arrival rate is low. According to measurement studies, the peer arrival rate for a file in BitTorrent is indeed low after the flash crowd phase. Therefore, in designing P2P file sharing systems, functioning of the system also in a case of a small population has to be taken into account.

Next in this chapter, by developing a detailed Markov chain model for two chunks, we were able to study the influence of the peer selection policy on the performance of the system. The policies studied in this thesis were random peer selection, random chunk selection, rarest chunk first selection and most common chunk first. In addition to the Markov model, we also used simulations to capture more complex systems.

Even though our model and simulations are limited, we can conclude that the biggest improvement in the mean lifetime is obtained when the file is divided from one piece into two pieces. With regard to the total download time, using 10 chunks improved the performance as compared to two chunks. Whether the number of the chunks is 10 or 100 does not have so significant influence on the performance anymore. One can assume that increasing the number of chunks also increases the overhead costs and delay. Taking this into account, one of our general conclusions from this chapter is that the optimal number of chunks, especially in terms of file availability, appears to be relatively small. As different peer selection policies are compared, our results suggest that none of the policies is clearly best. Instead, the superiority depends on the network parameters, such as the arrival rate of peers.

In Publication 10 we also briefly studied the stability of file sharing. Simulations showed that the stability of the system may be sensitive to the service time distribution, and the simple deterministic models presented in the literature are not adequate for finding stability bounds. Developing better approaches for studying stability issues is an interesting research question.

Finally, we proposed a spatio-temporal model for analyzing the resource usage of the P2P file sharing system in the underlying network. The analytical bounds for two different peer selection policies were derived. We found that by means of a policy where the closest peer is selected for download the resource usage of the network can be reduced to a fraction of the usage with random selection.

5 AUTHOR'S CONTRIBUTION

Publications 1, 2 and 3

These papers are joint works of the authors. First, Mr. Markus Peuhkuri performed the measurements in the Funet network and made the preliminary classification of the traffic into OD pairs. The present author and Mr. Ilmari Juva analyzed the traces and wrote the papers, except the introduction and the preliminary sections of Publication 1 and Publication 2, which were written by Dr. Samuli Aalto. In Publication 2, the present author was more responsible for classifying the traces and testing the Gaussian IID assumption, whereas Ilmari Juva analyzed rather the mean-variance relationship. In Publication 3 the present author studied the traffic volumes and diurnal variation of the OD pairs and Ilmari Juva the moving Gaussian IID model and mean-variance relationship.

Publication 4

This paper is a joint work of the authors. The algorithm was created together with all the authors and the paper was written by Dr. Samuli Aalto and the present author. The numerical evaluations were performed by the present author.

Publication 5

The idea of extending the adaptive algorithm to OSPF networks came from the present author, who also wrote the paper and made the numerical evaluations.

Publication 6

The idea of differentiating the traffic classes with WFQ-scheduling came from Prof. Jorma Virtamo, while the present author formulated the optimization problems, proposed approximations, and made the numerical evaluations.

Publication 7

This paper is an independent work by the author.

Publication 8

The idea of calculating the mean lifetime of the file sharing process came from the present author, while Prof. Jorma Virtamo gave the idea of the spatio-temporal model. The paper was written and the numerical examples given by the present author.

Publication 9

The ideas of this paper are mainly from the present author and it was also written by the present author, while Dr. Samuli Aalto instructed the work.

Publication 10

The main ideas and the modelling part of this paper was a joint work of both authors. The simulations were performed and the paper was written by the present author.

REFERENCES

[AA03] E. J. Anderson and T. E. Anderson. On the stability of adaptive routing in the presence of congestion control. In *Proceedings of IEEE INFOCOM*, 2003.

[AC03] D. Applegate and E. Cohen. Making intra-domain routing robust to changing and uncertain traffic demands, understanding fundamental tradeoffs. In *Proceedings of ACM SIGCOMM*, 2003.

[AMA+99] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. Requirements for traffic engineering over MPLS. IETF RFC 2702, September 1999.

[AW05] I. Akyildiz and X. Wang. A survey on wireless mesh networks. *IEEE Radio Communications*, 2005.

[AWK+99] G. Apostopoulos, D. Williams, S. Kamat, R. Guérin, A. Orda, and T. Przygienda. QoS routing mechanisms and OSPF extensions. IETF RFC 2676, August 1999.

[BDJT01] S. Bhattacharyya, C. Diot, J. Jetcheva, and Nina Taft. Pop-level and access-link-level traffic dynamics in a Tier-1 POP. In *Proceedings of ACM Internet Measurement Workshop (IMW)*, 2001.

[Ber98] D. Bertsekas. *Network Optimization: Continuous and Discrete Models*. Athena Scientific, 1998.

[BGG04] M. Barthelemy, B. Gondran, and E. Guichard. Spatial structure of the Internet traffic. *Physica A: Statistical Mechanics and its applications*, 319, 2004.

[BKT97] N.G. Bean, F. P. Kelly, and P. G Taylor. Braess' paradox in a loss network. *Journal of Applied Probability*, 1997.

[BRRT02] L. S. Buriol, M. G. C. Resende, C. C. Ribeiro, and M. Thorup. A memetic algorithm for OSPF routing. In *Proceeding of the 6th INFORMS Telecom*, 2002.

[But03] S. Butenweg. Two distributed reactive MPLS traffic engineering mechanisms for throughput optimization in best effort MPLS networks. In *Proceedings of IEEE Symposium on Computers and Communications*, July 2003.

[BVY03] P. Björklund, P. Värbrand, and D. Yuan. Resource optimization of spatial TDMA in ad hoc radio networks: A column generation approach. In *Proceedings of IEEE INFOCOM*, 2003.

[BZ97] J. Bennett and H. Zhang. Hierarchical packet fair queueing algorithms. *IEEE/ACM Transactions on Networking*, 5(5):675–689, 1997.

[Cac] Cachelogic. `http://www.cachelogic.com/home/pages/research/p2p2005.php`.

[CCLS01] J. Cao, W. Cleveland, D. Lin, and D. Sun. On the nonstationary of Internet traffic. In *Proceedings of ACM Sigmetrics*, 2001.

[CDWY00] J. Cao, D. Davis, S. Wiel, and B. Yu. Time-varying network tomography. *Journal of American Statistical Association*, 95:1063–1075, 2000.

[CFSD90] J. Case, M. Fedor, M. Schoffstall, and J. Davin. Simple network management protocol (SNMP). IETF RFC 1157, May 1990.

[Cla94] K. Claffy. *Internet Traffic Characterization*. PhD thesis, University of California, San Diego, 1994.

[CNRS98] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick. OSPF version 2. IETF RFC 2386, August 1998.

[Coh03] B. Cohen. Incentives build robustness in BitTorrent. In *Proceedings of First Workshop on Economics of Peer-to-Peer Systems*, 2003.

[DAJ00] E. Dinan, D. Awduche, and B. Jabbari. Optimal traffic partitioning in MPLS networks. In *Proceedings of Networking*, 2000.

[DKY03] K. Kompella D. Katz and D. Yeung. Traffic engineering (TE) extensions to OSPF version 2. IETF RFC 3630, September 2003.

[Duf04] N. Duffield. Sampling for passive internet measurement: A review. *Statist. Sci. 19*, (3):472–498, 2004.

[EJLW01] A. Elwalid, C. Jin, S. Low, and I. Widjaja. MATE: MPLS adaptive traffic engineering. In *Proceedings of IEEE INFOCOM*, 2001.

[ERP] M. Ericsson, M.G.G Resende, and P.M. Pardalos. A genetic algorithm for the weights setting problem in OSPF routing. *Journal of Combinatorial Optimization*, 6.

[FCL06] B. Fan, D-M Chiu, and J. Lui. Stochastic differential equation approach to model BitTorrent-like P2P systems. In *Proceedings of IEEE ICC*, 2006.

[FGL+01] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True. Deriving traffic demands for operational IP networks: Methodology and experience. *IEEE/ACM Transactions on Networking*, 9(3):265–279, June 2001.

[FLSC04] M. Feldman, K. Lai, I. Stoica, and J. Chuang. Robust incentive techniques for peer-to-peer networks. In *Proceedings of ACM Conference on Electronic Commerce (EC)*, 2004.

[FML+03] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and C. Diot. Packet-level traffic measurements from the Sprint IP backbone. *IEEE Network*, November 2003.

[FRT02] B. Fortz, J. Rexford, and M. Thorup. Traffic engineering with traditional IP routing protocols. *IEEE Communication Magazine*, 1, October 2002.

[FT00] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *Proceedings of IEEE INFOCOM*, 2000.

[FT02] B. Fortz and M. Thorup. Optimizing OSPF/IS-IS weights in a changing world. *IEEE Journal on Selected Areas in Communications*, 20(4), 2002.

[FTT+02] F. Faucheur, M. Tatham, T. Telkamp, J. Boyle, W. Lai, P. Hicks, A. Chiu, W. Townsend, and D. Skalecki. Requirements for support of diff-serv-aware MPLS traffic engineering. Internet draft <draft-ietf-tewg-diff-te-reqts-04.txt>, October 2002.

[FVFB05] M. Freedman, M. Vutukuru, N. Feamster, and H. Balakrishnan. Geographic locality of IP prefixes. In *Proceedings of ACM Internet Measurement Conference*, 2005.

[FWD+02] F. Le Faucheur, L. Wu, B. Davie, S. Davari, P. Väänänen, R. Krishnan, P. Cheval, and J. Heinänen. Multi-protocol label switching (MPLS) support of differentiated services. IETF RFC 3270, May 2002.

[GCX+05] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang. Measurement, analysis, and modeling of BitTorrent-like systems. In *Proceedings of USENIX Internet Measurement Conference*, 2005.

[GJT04] A. Gunnar, M. Johansson, and T. Telkamp. Traffic matrix estimation on a large IP backbone - a comparison on real data. In *Proceedings of ACM Internet Measurement Conference*, 2004.

[GK00] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, 2000.

[GKL+04] T. Güven, C. Kommareddy, R. La, M. Shayman, and B. Bhattacharjee. Measurement based optimal multi-path routing. In *Proceedings of IEEE INFOCOM*, 2004.

[GZRR03] I. Gojmerac, T. Ziegler, F. Ricciato, and P. Reichl. Adaptive multipath routing for dynamic traffic engineering. In *Proceedings of IEEE Globecom*, 2003.

[IL06] M. Ilvesmäki and M. Luoma. Characterizing Internet flows with aggregated TCP/UDP source port measurements. In *Proceedings of IPS-MOME*, 2006.

[IUKB+04] M. Izal, G. Uvroy-Keller, E.W. Biersack, P.A. Felber, A.Al Hamra, and L. Garcés-Erice. Dissecting BitTorrent: Five months in a torrent's lifetime. In *Proceedings of Passive and Active Measurement Workshop (PAM)*, 2004.

[JPPQ05] K. Jain, J. Padhye, V. Padmanabhan, and L. Qiu. Impact of interference on multi-hop wireless network performance. *Wireless Networks*, 11:471–487, 2005.

[JX06] M. Johansson and L. Xiao. Cross-layer optimization of wireless networks using nonlinear column generation. *IEEE Transactions on Wireless Communications*, 5(2), 2006.

[KBB+04] T. Karagiannis, A. Broido, N. Brownlee, K. Claffy, and M. Faloutsos. Is P2P dying or just hiding? In *Proceedings of IEEE Globecom*, 2004.

[KMFB04] T. Karagiannis, M. Molle, M Faloutsos, and A. Broido. A nonstationary Poisson view of Internet traffic. In *Proceedings of IEEE INFOCOM*, 2004.

[KN02] J. Kilpi and I. Norros. Testing the Gaussian aggregation of aggregate traffic. In *Proceedings ACM SIGCOMM Workshop on Internet Measurements*, 2002.

[KR05] B. Sikdar K.K. Ramachandran. Framework for modeling peer to peer networks. In *Proceedings of IEEE INFOCOM*, 2005.

[KYKV03] H. Kaur, T. Ye, S. Kalyanaraman, and K. Vastola. Network performance analysis of and adaptive OSPF routing strategy-effective bandwidth estimation. In *Proceedings of IEEE/ACM MASCOTS*, 2003.

[LCC+02] Q. Lv, P. Cao, E. Cohen, K. Li, and Scott Shenker. Search and replication in unstructured peer-to-peer networks. In *Proceedings of ACM International Conference on Supercomputing (ICS)*, 2002.

[LPC+04] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. Kolaczyk, and Nina Taft. Structural analysis of network traffic flows. In *Proceedings of ACM SIGMETRICS/Performance*, 2004.

[LPV06] P. Lassila, A. Penttinen, and J. Virtamo. Dimensioning of wireless mesh networks with flow-level QoS requirements. In *Proceedings of ACM PE-WASUN*, 2006.

[LR98] T. Li and Y. Rekhter. A Provider Architecture for Differentiated Services and Traffic Engineering (PASTE). IETF RFC 2430, October 1998.

[LTB03] W. Lai, R. Tibbs, and S. Berghe. Requirements for Internet traffic engineering measurement, July 2003.

[LTWW94] W. Leland, M. Taqqu, W. Willinger, and D. Wilson. On the self-similar nature of Ethernet traffic. *IEEE/ACM Transactions on Networking*, 2(1):1–15, 1994.

[LW91] W. Leland and D. Wilson. High time-resolution measurement and analysis of LAN traffic: Implication for LAN interconnection. In *Proceedings of IEEE INFOCOM*, 1991.

[MD00] S. Molnar and T. Dinh Dang. Pitfalls in long range dependence testing and estimation. In *Proceedings of the IEEE Globecom*, 2000.

[Moy98] J. Moy. OSPF version 2. IETF RFC 2328, April 1998.

[MRZ03] M. Thorup M. Roughan and Y. Zhang. Traffic engineering with estimated traffic matrices. In *Proceedings of IMC*, 2003.

[MTS+02] A. Medina, N. Taft, K. Salamatian, B. Bhattacharyya, and C. Diot. Traffic matrix estimation: Existing techniques and new directions. In *Proceedings of ACM SIGCOMM*, 2002.

[MV05] L. Massoulié and M. Vojnović. Coupon replication systems. In *Proceedings of ACM SIGMETRICS*, 2005.

[NhCR+03] T.S. Eugene Ng, Yang hua Chu, S. Rao, K. Sripanidkulchai, and H. Zhang. Measurement-based optimization techniques for bandwidth-demanding peer-to-peer systems. In *Proceedings of INFOCOM*, 2003.

[NK85] R. Nelson and L. Kleinrock. A collision-free multihop channel access control. *IEEE Transactions on Communications*, 22:934–944, 1985.

[Nor94] I. Norros. A storage model with self-similar input. *Queueing Systems*, 16(1):1–15, 1994.

[Nor95] I. Norros. The management of large flows of connectionless traffic on the basis of self-similar modeling. In *IEEE ICC*, 1995.

[OBC+01] T. Ott, T. Bogovic, T. Carpenter, K. R. Krishnan, and D. Shallcross. Algorithms for flow allocation for multi protocol label switching. Telcordia Technical Memorandum TM-26027T, 2001.

[Pax97] V. Paxton. *Measurement and Analysis of End-to-End Internet Dynamics.* PhD thesis, University of California, Berkeley, 1997.

[Peu01] M. Peuhkuri. A method to compress and anonymize packet traces. In *Proceedings of IMW*, 2001.

[PF95] V. Paxton and S. Floyd. Wide-area traffic: The failure of Poisson modelling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, 1995.

[PGES05] J.A. Pouwelsem, P. Garbacki, D.H.J. Epema, and H.J. Sips. The BitTorrent P2P file-sharing system: Measurements and analysis. In *Proceedings of IPTPS*, 2005.

[PKC96] K. Park, G. Kim, and M. Crovella. On the relationship between file sizes, transport protocols, and self-similar network traffic. In *Proceedings of the International Conference on Network Protocols*, 1996.

[PL02] T. Pereira and L. Ling. Network performance analysis of and adaptive OSPF routing strategy-effective bandwidth estimation. In *International Telecommunication Symposium ITS, Brazil*, 2002.

[QS04] D. Qiu and R. Srikant. Modeling and performance analysis of BitTorrent-like peer-to-peer networks. In *Proceedings of ACM SIGCOMM*, 2004.

[RD01] A. Rowston and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *Proceedings of Middleware*, 2001.

[RHKS02] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-aware overlay construction and server selection. In *Proceedings of INFOCOM*, 2002.

[RVC01] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. IETF RFC 3031, 2001.

[SBG+02] A. Sridharan, S. Bhattacharyya, R. Guérin, J. Jetcheva, and N. Taft. On the impact of aggregation on the performance of traffic aware routing. In *Proceedings of ITC 2002*, 2002.

[SGD03] A. Sridharan, R. Guérin, and C. Diot. Achieving near-optimal traffic engineering solutions for current OSPF-IS-IS networks. In *Proceedings of IEEE INFOCOM*, 2003.

[SKL+03] J. Song, S. Kim, M. Lee, H. Lee, and T. Suda. Adaptive load distribution over multipath in MPLS networks. In *Proceedings of IEEE ICC*, 2003.

[SLC01] Y. Seok, Y. Lee, and Y. Choi. Dynamic constrained multipath routing for MPLS networks. In *Proceedings of ICCCN*, 2001.

[SMLNDK03] I. Stoica, R. Morris, D. Liben-Nowell, and F. Dabek H. Balakrishnan D. Karger, M. Kaashoek. Chord: A scalable peer-to-peer lookup protocol for Internet applications. volume 11, pages 17–32, 2003.

[Spr00] S. Spraggs. Traffic engineering, carrier-scale IP networks. 18(3):235–260, 2000.

[TB86] J. Tsitsiklis and D. Bertsekas. Distributed asynchronous optimal routing in data networks. *IEEE Transactions on Automatic Control*, AC-31(4), 1986.

[TWN06] Y. Tian, D. Wu, and K. Ng. Analysis and improvement for BitTorrent-like file sharing networks. In *Proceedings of IEEE INFOCOM*, 2006.

[Uhl04] S. Uhlig. Non-stationary and high-order scaling in TCP flow arrivals: a methodological analysis. *ACM SIGCOMM Computer Communications Review*, 34(2), 2004.

[Var96] Y. Vardi. Network tomography: Estimating source-destination traffic intensities from link data. *Journal of American Statistical Association*, 91(433):365–377, 1996.

[vdMMP06] R. van de Meent, M. Mandjes, and Aiko Pras. Gaussian traffic everywhere? In *Proceedings of IEEE ICC*, 2006.

[Vil99] C. Villamizar. OSPF optimized multipath (OSPF-OMP). Internet draft<draft-villamizar-ietf-ospf-omp-02>, February 1999.

[War52] J. Wardrop. Some theoretical aspects of road traffic research. In *Proceedings of Inst. Civil Engineers*, 1952.

[WCZ+05] Y. Wu, P. Chou, Q. Zhang, K. Jain, W. Zhu, , and S. Kung. Network planning in wireless ad hoc networks: A cross-layer approach. *IEEE Journal on Selected Areas in Communications*, 23(1), 2005.

[WWZ01] Y. Wang, Z. Wang, and L. Zhang. Internet traffic engineering without full mesh overlaying. In *Proceedings of IEEE INFOCOM*, 2001.

[YdV04] X. Yang and G. de Veciana. Service capacity of peer to peer networks. In *Proceedings of IEEE INFOCOM*, 2004.

[Zha95] H. Zhang. Service disciplines for guaranteed performance service in packet-switching networks. *Proceedings of the IEEE*, 83(10), 1995.

[ZSZ+04] Z. Zhao, Y. Shu, L. Zhang, H. Wang, and O. Yang. Flow-level multipath load balancing in MPLS network. In *Proceeding of IEEE ICC*, 2004.

[ZZB05] K. Zu, Z-L Zhang, and S. Bhattacharyya. Profiling Internet backbone traffic: Behavior models and applications. In *Proceedings of ACM SIGCOMM*, 2005.