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

Riikka Susitaival

Adaptive Traffic Engineering in MPLS and OSPF Networks

This thesis has been submitted for official examination for the degree of Licentiate of Technology

Supervisor: Professor Jorma Virtamo, HUT

Instructor: Ph.D. Samuli Aalto

HELSINKI UNIVERSITY OF TECHNOLOGY Abstract of the Licentiate Thesis

Author:	Riikka Susitaival
Title of the thesis:	Adaptive Traffic Engineering in MPLS and OSPF Networks
Date:	November 3, 2004
Number of pages:	86

Department:	Department of Electrical and Communications Engineering
Professorship:	S-38 Networking Technology

Supervisor:	Professor Jorma Virtamo
Instructor:	Ph.D. Samuli Aalto

Traffic engineering refers to performance optimization of operational networks. One task of traffic engineering is load balancing, the idea of which 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. MPLS brings up new possibilities, such as explicit routing, for load balancing in IP networks. It has recently been found that a similar load balancing is possible to implement even in IP networks based on OSPF-routing by adjusting the link weights and the traffic splitting ratios in the routers. A common problem in traffic engineering mechanisms is that the traffic matrix is assumed to be known. This assumption is rarely true and thus adaptive approaches are needed.

In this thesis we study adaptive load balancing in both MPLS and OSPF networks based on measured link loads. We propose an adaptive and distributed algorithm for both types of networks that gradually balances the load by making small changes in the traffic splitting ratios of the paths or the routers. In addition, a numerical method is developed to evaluate the performance of the algorithm. The functioning of the algorithm is verified by numerical tests under different networks and traffic conditions. Also the effect of traffic fluctuations on the performance of the algorithm is evaluated.

The tests demostrate that in MPLS networks the adaptive algorithm converges rapidly almost to the optimum. In OSPF networks the convergence is slower, but we can still improve the performance of the network significantly as compared to equal splitting.

Keywords: MPLS, OSPF, load balancing, adaptive routing

Teknillinen korkeakoulu Lisensiaattityön tiivistelmä

Tekijä: Riikka Susitaival
Työn nimi: Adaptiivinen liikenteenhallinta MPLS- ja OSPF-verkoissa
Päivämäärä: 3.11.2004
Sivumäärä: 86

Osasto: Sähkö- ja tietoliikennetekniikan osasto
Professuuri: S-38 Tietoverkkotekniikka

Työn valvoja: Professori Jorma Virtamo
Työn ohjaaja: Fil. tri Samuli Aalto

Liikenteenhallinnalla voidaan optimoida tietoverkkojen suorituskykyä. Yksi liikenteenhallinnan tehtävistä on kuormantasaus, jossa ideana on hallitusti siirtää liikennettä kuormittuneilta linkeiltä verkon ruuhkattomampiin osiin. Jos tiedetään tarjottu liikenne, kuormantasaus voidaan muotoilla optimointiongelmana. Multi Protocol Label Switching (MPLS) -protokolla tuo mukanaan uusia mahdollisuuksia kuormantasaukseen IP-verkoissa. Äskettäin on huomattu, että samaan tulokseen voidaan päästä myös perinteisissä Open Shortest Path First (OSPF) -verkoissa linkkipainoja ja jakosuhteita muuttamalla. Yleinen ongelma näissä kuormantasausmenetelmissä kuitenkin on, että liikennematriisi oletetaan tunnetuksi. Oletus on useimmiten epärealistinen ja siksi tarvitaan adaptiivisia liikenteenhallintamenetelmiä.

Tämän työn tarkoituksena on tutkia kuormantasausta sekä MPLS- että OSPF-verkoissa. Työssä kehitetään adaptiivinen ja hajautettu algoritmi, joka tasaa kuormaa tekemällä asteittaisia muutoksia liikennejakaumassa jakosuhteita muuttamalla. Tämän lisäksi kehitetään numeerinen menetelmä algoritmin suorituskyvyn arvioimiseksi ja suoritetaan numeerisia kokeita algoritmin toiminnan testaamiseksi. Myös tarjotun liikenteen vaihtelun vaikutusta algoritmin suorituskykyyn tutkitaan.

Yhteenvetona voidaan sanoa, että adaptiivinen algoritmi konvergoi lähes optimiin MPLS-verkoissa. OSPF-verkoissa konvergenssi ei ole yhtä nopeaa, mutta suorituskyvyn parantuminen tasajakoon verrattuna on silti merkittävää.

Avainsanat: MPLS, OSPF, kuormantasaus, adaptiivinen reititys

Preface

The work for this licentiate thesis was carried out in the Networking Laboratory of Helsinki University of Technology as a part of the COST 279, IRoNet and FIT projects. I have also got funding from GSNIS graduate school.

I would like to thank my supervisor, professor Jorma Virtamo, for his valuable guidance and ideas. I would also like to thank my instructor Samuli Aalto for his dedicated work and help in the scientific writing process. I would also thank my colleagues in the Networking Laboratory for friendly atmosphere and relaxation at the coffee breaks.

Finally, I would like to thank my family and boyfriend Teemu for the support I have got during my studies.

Espoo, 3.11.2004,

Riikka Susitaival

Contents

Acronyms

- **AMP** Adaptive Multiple-Path
- **AS** Autonomous System
- **ATM** Asynchronous Transfer Mode
- **BA** Behavior Aggregate
- **BGP** Border Gateway Protocol
- **BM** Backpressure Message
- **BTT** Bidirectional Traffic Trunk
- **CR-LDP** Constraint Routing Label Distribution Protocol
- **ECMP** Equal Cost Multiple Path
- **EGP** Exterior Gateway Protocol
- **EM** Expectation Maximization
- **EWMA** Exponentially Weighted Moving Average
- **FEC** Forwarding Equivalence Class
- **GMPLS** Generalized Multi Protocol label Switching
- **G-P** Gradient-projection
- **ICMP** Internet Control Message Protocol
- **IETF** Internet Engineering Task Force
- **IGP** Interior Gateway Protocol
- **IE** Ingress-egress
- **ILM** Incoming Label Map
- **IP** Internet Protocol

- **IPMON** IP Monitoring
- **IS-IS** Intermediate System-Intermediate System
- **ISP** Internet Service Provider
- **LAN** Local Area Network
- **LDP** Label Distribution Protocol
- **LDM** Load Distribution over Multipath
- **LMP** Label Management Protocol
- **LP** Linear Programming
- **LSA** Link State Advertisement
- **LSP** Label Switching Path
- **LSR** Label Switching Router
- **NHLFE** Next Hop Label Forwarding Entry
- **MAM** Maximum Allocation Multiplier
- **MATE** MPLS Adaptive Traffic Engineering
- **MIB** Management Information Base
- **MIP** Mixed Integer Programming
- **MPLS** Multi Protocol Label Switching
- **MPLS-OMP** MPLS Optimized Multipath
- **NLP** Non-Linear Programming
- **OD** Origin-destination
- **OSPF** Open Shortest Path First
- **OXC** Optical Cross-connect
- **QoS** Quality of Service
- **RFC** Request For Comments
- **RIP** Routing Information Protocol
- **RSVP** Resource Reservation Protocol
- **SNMP** Simple Network Management Protocol

- **SPSA** Simultaneous Perturbation Stochastic Approximation
- **TCP** Transmission Control Protocol
- **TE** Traffic Engineering
- **TLV** Type/Length/Value
- **TOS** Type Of Service
- **UDP** User Datagram Protocol
- **VC** Virtual Circuit
- **WDM** Wavelength-Division Multiplexing

Chapter 1

Introduction

1.1 Background

During the last decade the Internet has expanded to a world-wide network connecting millions of hosts and users. The fundamental idea of the Internet was to serve the customers on best-effort basis by a simple and robust Internet Protocol (IP). In developing new services, especially those involving real time traffic, Quality of Service (QoS) has become more and more important in network operation. For this reason QoS architectures, such as Integrated Services (IntServ) and Differentiated Services (DiffServ) have been developed and have gained a lot of attention. QoS routing mechanisms try to select routes in such way that performance objectives are satisfied.

Many emerging applications, such as peer-to-peer applications, are bandwidth intensive in their nature and thus efficient use of network resources necessitates implementation of some traffic engineering mechanisms. 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 scale network control to network planning at a longer time period. Common to all the mechanisms is the aim to alleviate congestion of the network and thus to improve the satisfaction of the customers. The term *load balancing* refers to the congestion avoidance mechanism where traffic is moved from congested links to less loaded areas of the network.

In traditional IP networks the tools for traffic engineering are limited due to the lack of control of the used routes. One purpose of Multi Protocol Label Switching (MPLS) is to provide more capabilities to TE. 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 the interest to develop similar traffic engineering mechanisms in the shortest-path networks has re-emerged. Many algorithms have been developed to optimize the link weights, which determine the paths in use. Using these algorithms the performance of the network can be improved almost to the same level as with MPLS traffic engineering.

For developing efficient traffic engineering mechanisms the information on traffic offered to the network is crucial. Many proposed traffic engineering mechanisms assume that the traffic demands are known. However, defining the traffic matrix describing the point-to-point demands in the network is not a straightforward task.

When the traffic demand matrix is not available or changes unpredictably, adaptive traffic engineering mechanisms are required. One way to develop a traffic-aware routing mechanism is to first measure the link loads, which is quite easy using wide-spread management protocols such as SNMP, and then take appropriate control actions.

1.2 Objectives

The objective of this thesis is to study two aspects of traffic engineering in an autonomous system (AS). The first aspect is related to the architecture in which traffic engineering functionalities are performed, such as the connection-oriented MPLS network and the connectionless shortest path network. In the second aspect, the traffic engineering mechanisms are categorized into static and adaptive, the former one trying to find an optimal load distribution in the network, whereas in the latter approach the traffic allocation is continuously adjusted based on the traffic conditions, such as measured link loads.

Specifically, we develop for both MPLS and OSPF networks adaptive and distributed load balancing algorithms that make incremental changes in the load distribution based on the link load measurements. We also describe a numerical method for evaluating the algorithms. By this method the algorithms are evaluated in different test-networks and under various traffic scenarios. The purpose is also to compare the benefits and difficulties of traffic engineering in MPLS and OSPF networks.

1.3 Structure of the thesis

The thesis is organized as follows: In Chapter 2 we introduce traffic engineering in depth. We formulate the static load balancing problem with various objective

functions and then consider adaptive traffic engineering and the requirements set to adaptive algorithms.

In Chapter 3 traffic measurements related to traffic engineering are considered. Some passive and active methods are introduced. Also the problem related to infering the traffic matrix is discussed.

Chapter 4 covers the traffic engineering related to MPLS networks. First the technical issues of MPLS architecture are introduced. Then we review both static and adaptive load balancing algorithms. In the last section we develop an adaptive algorithm for balancing the load and evaluate its performance in many test-networks and under various traffic conditions.

In Chapter 5 we introduce shortest-path routing protocols, especially the OSPF protocol. Then the main advances of TE in OSPF networks are reviewed. Both static and adaptive methods are considered in separate sections. Finally, we apply the adaptive algorithm of Section 4 to OSPF networks and explore its performance extensively.

In Chapter 6 we compare the results of previous two sections. Pros and cons of MPLS-and OSPF-based load balancing are discussed. Finally, in Chapter 7 we conclude the thesis and discuss future research topics.

1.4 Related publications

An essential part of the results presented in this thesis have been published in [Sus04a] and [Sus04b]. In [Sus04a] we consider adaptive load balancing in an MPLS network, which is covered in Section 4.4. A study of load balancing in the OSPF context, the content of Section 5.5, has been published in [Sus04b]. In addition, Section 4.2.2 briefly reviews the results related to solving the static load balancing problem, which are published in [Sus02]. As compared with the publications mentioned, this thesis tries to give a broader view on the subject by reviewing related research and propositions in more detail, and by presenting more numerical results and graphs based on our own contribution.

Chapter 2

Traffic engineering

This chapter introduces the concept of traffic engineering. We state the objectives of traffic engineering by formulating the static load balancing problem. Different aspects related to adaptive traffic engineering are also introduced.

2.1 Introduction to Internet traffic engineering

Traffic engineering refers to the optimization of operational networks. The idea is to use existing resources most effectively, which is complementary to dimensioning, where one adds resources to meet the requirements for serving customers properly. On one hand, we have a network which has a limited capacity, on the other hand the customers who offer traffic to the network and have to be served sufficiently. Both capacity and traffic demands vary at a long time scale as the operators build new infrastructure and the customer demands change along 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 the applications with various traffic characteristics differently. Traffic engineering and QoS are strongly related, but to make a difference, QoS controls how the resources are allocated to different users, whereas TE controls where in the network resources are used.

Performance optimization of IP networks can be done both at the traffic level and at the resource level. Traffic oriented performance concentrates on the quality of service of traffic streams. Minimization of packet loss and delay, maximization of throughput and execution of service level agreements are the major measures to be improved. The resource oriented performance objectives consist of efficient

resource utilization and resource management. Usually bandwidth is the most scarce resource, so a major function is to manage bandwidth allocation.

Both from traffic and resource oriented perspectives the minimization of congestion is crucial. 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 network is over-utilized while another part of network has unused bandwidth. The second type of congestion can be effectively alleviated using techniques provided by traffic engineering such as load balancing policies. The objective can then be minimizing the maximum link utilization. However, traffic engineering should be carried out in such a way that congestion can be managed cost-effectively.

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 modification of traffic management parameters, modification of routing parameters and modifications of resource attributes and constraints [RFC2702].

In the current IP networks the most common Interior Gateway Protocols (IGPs) are Open Shortest Path First (OSPF) and Intermediate System-Intermediate System (IS-IS) [Gha99]. Traffic is routed along the shortest path since the use of resources is then minimized. However, IGPs that are based on shortest path calculations do not take into account the congestion avoidance. Algorithms optimize the paths based on simple metrics but the adequacy of bandwidth is not considered.

Equal Cost Multipath (ECMP) refers to a mechanism in which traffic is split equally to the parallel shortest paths from a router to a given destination [RFC2328]. The support of ECMP improves performance in a situation where a traffic stream is routed over a link that does not have enough bandwidth. However, 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 the large networks.

The inadequacies of IGPs can be overcome by the overlay models like IP over ATM and IP over Frame Relay. These models construct virtual topologies from virtual circuits (VC) and then provide services like constraint-based routing at the VC level. Nevertheless, these techniques have also scalability problems. MPLS is a new routing architecture that provides attractive tools to realize the traffic engineering in IP networks flexibly.

We depict a framework, based on the model presented in [For02b], for traffic engineering in Figure 2.1. In this model, the topology and traffic demands derived from the operational network act as input parameters for traffic engineering. Based on these parameters an optimal way to carry traffic over the network

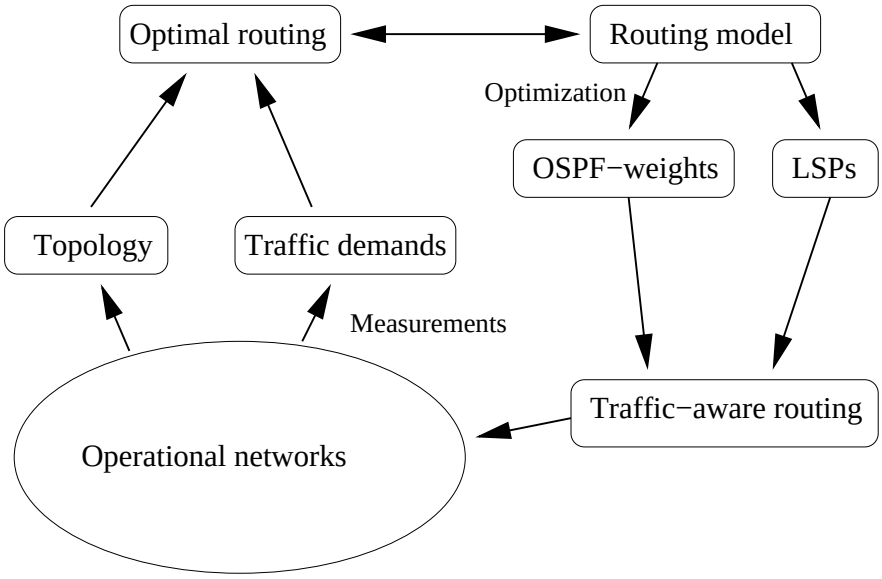

Figure 2.1: Traffic engineering model

is defined. Routing model refers to the underlying routing protocol, such as OSPF or MPLS. The 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 of MPLS can be determined based on 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 into the network with given parameters. On the other hand, the routing model sets constraints to the optimal traffic allocation (use of the shortest paths, for example). As a result, we have traffic-aware routing, which should provide better performance than a 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, the admission control mechanisms are outside of the scope of this thesis.

Different functionalities of traffic engineering can be categorized based on the time scale at 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 as capacity planning and routing design at the time scale of days to weeks. Finally, network planning is responsible for predicting the long-term demand and the replacement of the nodes at the time scale of months and years.

2.2 Input parameters for traffic engineering

In this section we study the input parameters of traffic engineering, topology and traffic demands, in more detail. It should be noted that the traffic engineering framework of Figure 2.1 is a simplified version of the whole system. There are many other factors that affect on the success of traffic engineering, such as the type of applications loading the network.

2.2.1 Topology

In order to make clever routing decisions we need a view of the state of the network. Topology as an input parameter consists of information on routers (also referred to as nodes) and links and their capacities. This information can be obtained from router configuration data. The Simple Network Management Protocol (SNMP) provides also information of the state of different network elements [Fel01]. Topology can usually be taken as fixed because the changes in router and link capacities and their replacements occur at a long time scale.

2.2.2 Traffic demands

In the network planning and in the evaluation of the effects of the changes in routing parameters, a network-wide view of traffic is crucial. The local traffic conditions, such as link loads, are just the result of the routing decisions already made in the network.

Three models to represent the traffic volumes that travel across the network are presented in [Gro02]. The first one is *path matrix* that defines the data volumes for every path between every source and destination node (see the left side of Figure 2.2). Typically, the path matrix is a result of the current routing. For example, in an MPLS network, the paths between node pairs can be defined explicitly and the path matrix is can be derived by measurements.

The second model is *traffic matrix*, which represents offered load between each ingress and egress pair. This model is depicted in the middle of Figure 2.2. Traffic matrix is widely used in many research papers as input for traffic engineering.

Often the terms ingress-egress and origin-destination are confused. To make it clear, ingress and egress nodes refer to edge nodes of a routing domain, whereas origin and destination nodes are related the end nodes of route from the source host to the destination host and are defined by the IP address. In this thesis we concentrate on the traffic engineering mechanisms that models the traffic of the network by the traffic matrix, especially in one autonomous routing domain.

Finally, the third concept is *demand matrix* that describes the traffic volumes between an ingress node and the set of egress nodes (see right side of Figure 2.2). According to [Gro02], the use of the demand matrix is reasonable in traffic engineering of IP networks since, as traffic is routed to neighboring routing areas, BGP routing selects a set of egress nodes for a given destination prefix instead of single point. The actual egress node is selected from the candidates by intradomain routing protocol.

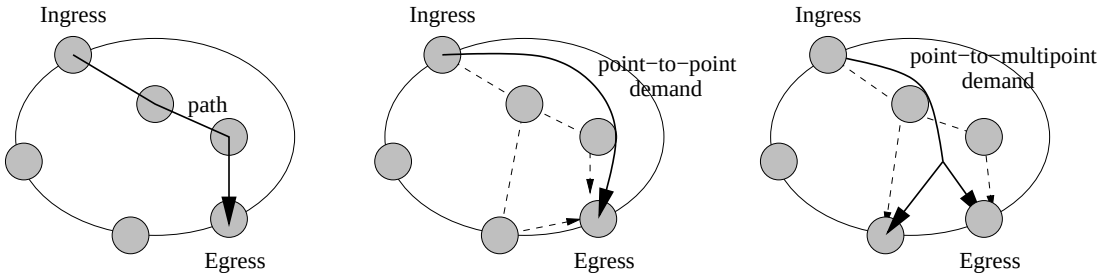

Figure 2.2: Traffic models: Left: path matrix. Middle: traffic matrix. Right: demand matrix

Another issue is how network-wide traffic models explained above are constructed in operational IP networks. The fact is that information is not directly available. Different estimation and monitoring methods have to be used instead. A question about traffic matrix estimation is discussed in more detail in Section 3.5 together with other measurement issues.

2.3 Static traffic engineering

In this section we study the static traffic engineering by formulating a classic load balancing problem and introducing different objective functions for that. We also introduce one algorithm, namely the gradient-projection algorithm, for solving the static load balancing problem.

2.3.1 Formulation of static load balancing problem

In this section we review the well-known static load balancing problem[1] [Ber92]. In this problem the offered traffic of each ingress-egress pair is carried in the network in such a way that some objective function is minimized. Depending on application, the paths used by each IE-pair can be arbitrary or selected beforehand. First we consider an optimization problem with unconstrained paths and then a problem with the predefined set of paths.

[1]Also known as optimal routing problem.

Paths unconstrained

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; $x^k \in \mathbb{R}^{L \times 1}$, $k \in \mathcal{K}$, refers to the link load vector with elements x_l^k; and $R^k \in \mathbb{R}^{N \times 1}$, $k \in \mathcal{K}$, denotes the vector for which $R_{s_k}^k = d_k$, $R_{t_k}^k = -d_k$, and $R_n^k = 0$ otherwise.

The rate of traffic allocated by IE-pair k on link l is denoted by x_l^k. These rates are the control variables in our static load balancing problem. Given the x_l^k, the induced load y_l on link l is

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

It is easy to see that, in general, the mapping between the traffic demands d_k and the link loads y_l is not one-to-one, i.e. while d_k's determine y_l's uniquely, the opposite is not true. Load y_l on link l incurs cost $C_l(y_l)$, which is assumed to be an increasing and convex function of the load y_l.

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

$$\begin{aligned}
&\text{Minimize} \quad C(x) = \textstyle\sum_{l \in \mathcal{L}} C_l(y_l) \\
&\text{subject to the constraints} \\
&x_l^k \geq 0, && \text{for each } l \in \mathcal{L} \text{ and } k \in \mathcal{K} \\
&\sum_{k \in \mathcal{K}} x_l^k \leq b_l, && \text{for each } l \in \mathcal{L}, \\
&Ax^k = R^k, && \text{for each } k \in \mathcal{K},
\end{aligned} \tag{2.1}$$

In this problem we have three constraints, the first one states that link loads should be positive, 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, the load balancing gives the linkwise portions x_l^k of the original demands d_k but the routes for these demands through the network are not necessarily unique. However, the routes are guaranteed to be loop-free.

Paths predefined

Now we consider the case in which the paths are defined explicitly as in the MPLS networks. Each IE-pair k has a predefined set $\mathcal{P}_k$ of the 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

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

Also in this scenario, the mapping between the traffic demands d_k and the link loads y_l is not one-to-one.

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:

$$\begin{aligned}
&\text{Minimize} \quad C(x) = \textstyle\sum_{l\in\mathcal{L}} C_l(y_l) \\
&\text{subject to constraints} \\
&x_p \geq 0, && \text{for all } p \in \mathcal{P}, \\
&y_l \leq b_l, && \text{for all } l \in \mathcal{L}, \\
&\textstyle\sum_{p\in P_k} x_p = d_k, && \text{for all } k \in \mathcal{K}.
\end{aligned} \tag{2.2}$$

As in problem (2.1), the pathwise solution is not necessarily unique when the paths share common links. If path set P contains all possible paths between the ingress and egress nodes, the constrained static load balancing problem is similar to problem (2.1).

2.3.2 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.

Linear problem

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 is then

$$C_l(y_l) = a_l y_l \quad \text{for all } l \in \mathcal{L}. \tag{2.3}$$

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

Minimization of the mean delay

Another frequently used link cost is the following nonlinear function:

$$C_l(y_l) = \begin{cases} \dfrac{y_l}{b_l - y_l}, & y_l < b_l, \\ \infty, & y_l \geq b_l. \end{cases} \tag{2.4}$$

With this choice, the total cost function $C(x)$ in problems (2.1) and (2.2) 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 link can be modelled as an M/M/1 queue and therefore the network as a Jackson queueing network. Although this assumption is not necessarily realistic 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 [For00], optimization of a non-linear problem is considered be too time-consuming. Thus it is possible to approximate the optimization problem with piece-wise linear optimization which can be solved optimally in polynomial time. Minimization of network delay also consumes more resources than the linear problem presented in the previous section. In some situations this can cause undesirable effects on the performance of the network.

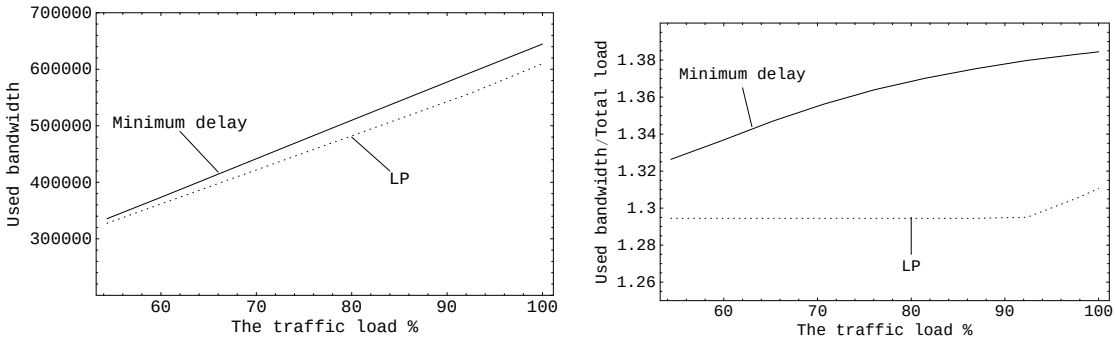

Figure 2.3: Used bandwidth on the left side and used bandwidth divided by the total offered traffic on the right side as a function of the traffic load.

As an example, we compare in Figure 2.3 the usage of bandwidth when traffic is routed by the linear optimization (objective function (2.3)) with link costs equal to 1 and by the network delay minimization (objective function (2.4)). In the linear optimization the shortest paths are selected if there is enough capacity. When the traffic load is over 90 %, the shortest paths are full and longer paths have to be utilized. When the mean delay is minimized, also with lighter load longer paths are used in order to balance the load, which can be seen from the greater bandwidth usage. The example network consists of 10 nodes, 52 links and 72 IE-pairs.

2.3.3 Alternative formulation of the static load balancing problem

In another formulation of the static load balancing problem the maximum link cost is minimized instead of the total cost. The problems (2.1) or (2.2) are replaced by following:

$$\begin{aligned} &\text{Minimize} \quad C(x) = \max_{l \in \mathcal{L}} C_l(y_l) \\ &\text{subject to constrains as in (2.1) or (2.2)} \end{aligned} \tag{2.5}$$

In this case a typical cost function is

$$C_l(y_l) = \frac{y_l}{b_l}. \tag{2.6}$$

By this choice the maximum link utilization is minimized. Computational studies have shown that the result of this objective is very close to the result of the mean delay minimization. Thus non-linear delay minimization problem can be

approximated by linear min-max problem introduced in this section and therefore computational efforts are spared. We note that the minimization of the maximum link utilization may result in long paths consuming much resources, while the minimization of the total mean delay both balances the load and gives preference to short paths.

The paper [Ott01] defines that the routing that allocates traffic load x_l on link l is *non-dominated* if there does not exist another routing with link-loads x'_l with property

$$x'_l \leq x_l \text{ for all } l \text{ and } x'_l < x_l \text{ for at least one } l. \tag{2.7}$$

In other words, non-dominance means that it is not possible to find another optimal solution that uses less capacity. The problem of minimization of maximum link utilization (2.5) and (2.6) has the disadvantage that an optimal solution is not necessarily non-dominated. If there exists a bottleneck in the network which determines the optimal solution, the rest of the network remains unbalanced. That is the reason why in many papers as [Ott01] and [Wan01a] a combined parametric criterion is presented . This LP-problem minimizes the maximum link utilization with a greater weight but also takes into account the overall usage of the resources with a smaller weight. The problem for unconstrained path set is as follows:

$$\begin{aligned}
&\text{Minimize } \alpha + r \sum_{k \in \mathcal{K}} \sum_{l \in \mathcal{L}} x_l^k \\
&\text{subject to the constraints} \\
&\alpha \geq 0, \\
&x_l^k \geq 0, && \text{for each } k \in \mathcal{K} \text{ and } l \in \mathcal{L}, \\
&\sum_{k \in \mathcal{K}} x_l^k \leq \alpha b_l, && \text{for each } l \in \mathcal{L}, \\
&Ax^k = R^k, && \text{for each } k \in \mathcal{K},
\end{aligned} \tag{2.8}$$

where α is a free variable describing the minimum of the maximum link utilization and r is some small constant.

In the case where the path set is predefined the combined optimization problem is:

$$
\begin{aligned}
&\text{Minimize } \alpha + r \sum_{k \in \mathcal{K}} \sum_{l \in \mathcal{L}} \sum_{p \in P_k : l \in p} x_p \\
&\text{subject to the constraints} \\
&\alpha \geq 0, \\
&x_p \geq 0, && \text{for all } p \in \mathcal{P}, \\
&\textstyle\sum_{k \in \mathcal{K}} \sum_{p \in P_k : l \in p} x_p \leq \alpha b_l, && \text{for all } l \in \mathcal{L}, \\
&\textstyle\sum_{p \in P_k} x_p = d_k, && \text{for all } k \in \mathcal{K}.
\end{aligned}
\tag{2.9}
$$

2.3.4 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 the communication networks.

Gradient-projection algorithm

A gradient-projection algorithm for the static load balancing problem with a predefined path set is presented in [Tsi86]. 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:

$$C_p = \sum_{l \in p} C_l(y_l). \tag{2.10}$$

The first derivative length of this path is

$$C'_p = \sum_{l \in p} C'_l(y_l), \tag{2.11}$$

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 2.3.1).

Let $\bar{p}_k$ denote the path of the IE-pair k which has the smallest first derivative length (2.11). 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:

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

where H is an approximation of the Hessian matrix,

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

α_H is a scaling factor and $C_l''(y_l)$ 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,

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

2.4 Adaptive traffic engineering

In this section we consider adaptive traffic engineering in general. The history on adaptivity in the Internet is discussed, as well as the requirements set for the adaptive algorithms.

2.4.1 Adaptivity in general

Adaptive traffic engineering refers to a system in which network control parameters are automatically adjusted based on the detected changes in 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 failure. By parameter adjustments one can have influence on the routes, traffic splitting and scheduling, and thus the performance of the network can be improved.

Figure 2.4 depicts an example of the adaptive system, in which the state of the network is measured regularly and control actions to meet the specified performance or the cost requirements are done based on the measurement information. These decisions are then forwarded to the network elements to obtain the desired changes in the network parameters.

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

Already in the ARPANET there were attempts to optimize the use of resources dynamically by changing the link metrics, which define the paths for the traffic incoming 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. A general assumption has for a long time been that it is too difficult to design a load-sensitive routing, which is both sensitive and does not generate too much overhead.

The answer to the question whether an adaptive network control is necessary at all depends largely on the network conditions, that is, the amount of capacity available in comparison with the traffic. The task of dimensioning is to ensure that there is always enough capacity in the network to serve the demand. This principle is also referred to as overprovisioning. Overprovisioning applies for the most of the links (according to [Fra03] most of the links are utilized under 50%), but recent studies show that there exist spikes on some links, in which the link utilization is over 90% [And03]. Therefore the use of adaptive mechanisms is

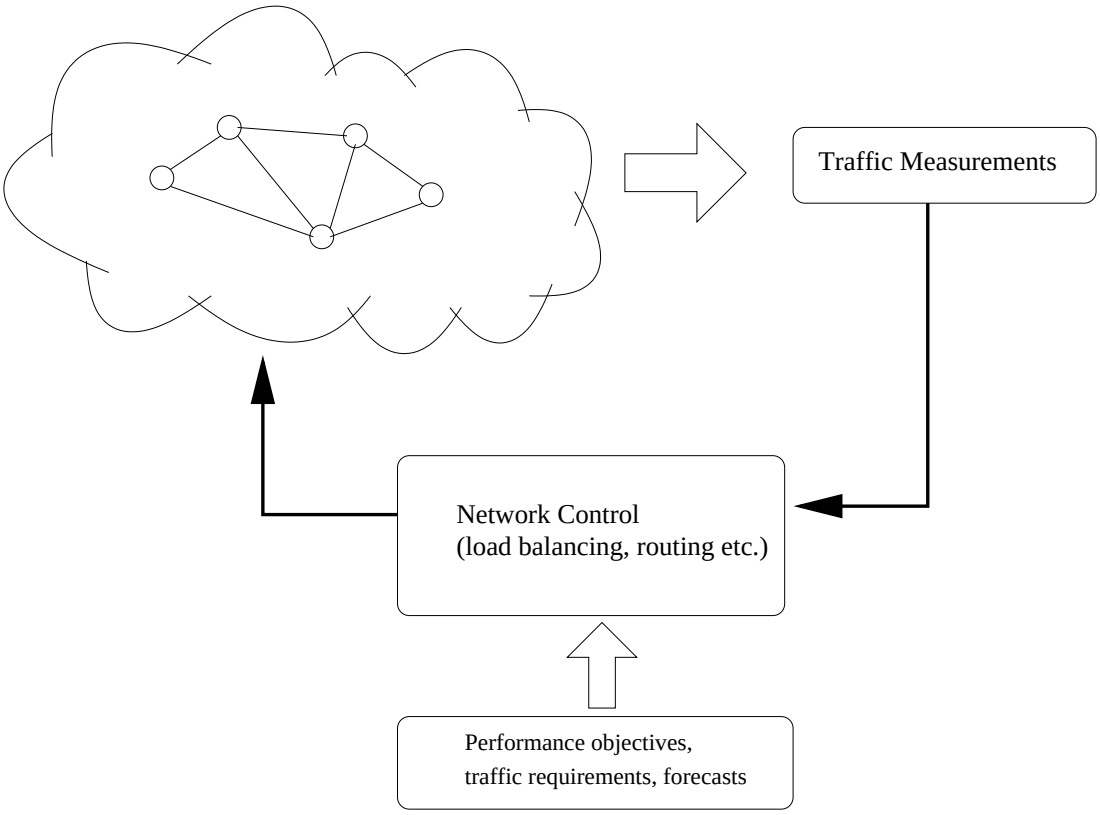

Figure 2.4: Adaptive traffic engineering model

reasonable.

In some cases where a large population of users can affect the used routes, adaptivity is a natural part of the system. If the user experiences that the currently used route is congested, it can select another one and thus traffic load is gradually balanced. The Wardrop equilibria defined in the context of transportation networks [War52] states that in the equilibrium state the journey times of the used paths are equal and shorter than the journey times of unused paths. However, the equilibrium does not provide that the obtained stable solution is optimal. Indeed, a counter-example is shown in [Bea97], where the addition of an extra link surprisingly decreases the performance of the loss network.

2.4.2 On the goodness of adaptive algorithms

There are many factors that have an influence on the success of the adaptive algorithms. On one hand, technical aspects, such as the amount of overhead and cost of additional hard- and software, may restrict a wide use of the algorithms. On the other hand, the algorithms themselves have to satisfy certain requirements, such as stability and fast convergence. In [And03] Anderson and Anderson dis-

cuss the stability and convergence of the adaptive algorithms, specially in the presence of congestion control.

In short, stability means that the algorithm converges to a good solution and, after that, the system remains unchanged until traffic conditions change. Properties of a stable system listed in [And03] are:

1. The system converges to some stable state: If the input, such as traffic load, remains unchanged, there should not be large fluctuations in the system behavior.

2. The state to which the system converges, should be close to the optimum, that is, it should provide a good performance. This is an important property since a stable routing without convergence requirements to the optimum is trivial. Consider routing along the paths that minimize the hop count as an example.

3. As the system has converged, the route oscillations should be eliminated (routing table changes, for example).

4. The three properties above should be independent of the topology of the network.

It is obvious that the rate of convergence should be as fast as possible. For example, if the time scale of adaptation is longer than the time scale of traffic demand changes, adaptive routing is unpractical. As a conclusion, to overcome prevailing skepticism existing against adaptive routing, the stability issues have to be addressed properly. Many times a stable non-optimal solution is preferable to one with oscillations around optimality.

Chapter 3

Traffic measurements for traffic engineering

The goal of this chapter is to provide a view to network measurements that are an essential part of a traffic engineering system. We discuss both passive and active methods. The problem of traffic matrix estimation is also described.

3.1 Introduction to Internet Measurements

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

The measurements can be classified in many ways. Location of measurement indicates which part or element of the network is measured and what exactly is 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 or 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 [Lai01]. Traffic characterization refers to the identification of traffic patterns, their distributions and the trends in the network evolution. Network monitoring determines the state of the network, such as the fault of some network element, and observes the quality of network services. The purpose of traffic control is to react to changes in the

network state at the short time scale by mechanisms such as rerouting, resource reservation and admission control.

3.2 What can be measured?

Regardless of the used technique, the measurements can be classified based on where and at which level of aggregation the measurements are performed. This is also referred to as *measurement base*. The measurement bases listed in [Lai01] are:

1. Flow-based: a detailed information about a flow collected usually on interfaces at access routers, edge routers or aggregation routers. This information consists of source and destination IP addresses/port numbers, type of service etc.
2. Interface-based, link-based, node-based: passive measurements at a single network element including information about packet counts, packet discards and errored packets. SNMP monitoring is an example of these measurements.
3. Node-pair-based: normally active measurements used to determine the performance between two edge routers.
4. Path-based: ability of MPLS to predefine the paths for traffic enables also path-based measurements.

Measurement entity describes what is measured. The combination of measurement base and entity defines a measurement type. Some common measurement entities, which are related to traffic characteristics and performance of the network, are listed in [Lai01]:

1. Traffic volume: number of bits/bytes/packets counted over a given time period at different aggregation levels.
2. Average holding time: flow lifetime or duration of an MPLS path.
3. Available bandwidth of a link or path: can be used in load balancing or admission control.
4. Throughput: rate of traffic (number of bits/bytes/packets per second) excluding loss.
5. Delay and delay variation.

6. Packet loss.

7. Resource usage: link/router utilization, buffer occupancy.

3.3 Passive measurement methods

Simple Network Management Protocol (SNMP) standardized by IETF in [RFC1157] is a widely-spread management protocol of today's Internet. SNMP defines how management information is stored and how communication between different elements of the network is performed. The network consists of network management stations and agents located at the network elements that perform the management functions assigned by management stations. The management information is stored in Management Information Base (MIB), which is tried to be kept as simple as possible.

MIB-II is the most common standardization of MIB and provides coarse-grained statistics about the network elements such as the status information of the interfaces or counts of packets passed through a network interface. Typically, the SNMP measurement with MIB-II provides the average traffic load of a link calculated over 5 minutes period. A more fine-grained and complicated statistics can be achieved by the MIB called RMON, which performs a remote monitoring of Local Area Networks (LANs). As a conclusion, MIB-II is a suitable management system for monitoring the network state and traffic variables such as link utilization at the slow time scale (minutes to hours). RMON provides more flexibility and detailed information but performs badly in large backbone networks [Gro02].

Packet monitoring refers to an approach in which a monitor makes 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 information. Thus the amount of copied and analyzed data has to be reduced by focusing on some field in the packet such as the IP header or using sampling. Nevertheless, detailed information produced by packet monitoring can be very useful in the network management.

Fraleigh et al. [Fra03] 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 from the period of several days on the backbone links, mark each packet with a submicrosecond timestamp, and synchronize timestamps so that the maximum error between stamps is 5 μs. The results of extensive measurements show many new characteristics of IP traffic. First, since SNMP provides the average load measured over 5 minutes interval, it does not detect short term congestion. In IPMON the measurement period is only 1 s and

thus it is possible to find that traffic is much more bursty in a finer granularity. It is also found that 60% of the Internet traffic is generated by new applications such as file sharing and media streaming and only 30% is Web traffic.

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 after that 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. Information on the link loads is distributed to the other routers in the network by the flooding mechanism of some routing protocol, such as OSPF. The measurement period in the system is several minutes.

3.4 Active methods

Active measurements are performed by sending probe packets and analyzing 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.

3.5 Traffic matrix estimation

In managing operational networks it is important to have a network-wide view of the traffic. Indeed, many traffic engineering mechanisms assume that a 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 measures indicate only the decisions made by the current routing protocol, that is, the effects of traffic demands. To find out the offered load per OD-pair different forecasts and estimates have to be used instead.

There are four main approaches to derive the traffic matrix [For02b]. 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 to 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.

The earliest study of deriving traffic demands in an IP backbone was presented by Feldmann et al. in [Fel01]. Instead of estimating point-to-point traffic demands, according to the paper, it is more preferable to model point-to-multipoint traffic volumes, which fit better to the nature of IP routing (see an example in the right side of Figure 2.2). An ISP cannot control the ingress link of traffic but the choice of the egress link depends both on intradomain and interdomain routing parameters and thus point-to-point traffic demands depend on the routing decisions. The traffic demands in [Fel01] 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.

Recently a lot of research effort has been used to find techniques to estimate the traffic matrix from the link load measurements and the static routing parameters such as the link weights. Term *network tomography* refers to the statistical inference methods that use repeated link load measurements in the traffic matrix estimation. The link counts are obtained from the SNMP data and the paths by calculating the shortest paths between each OD-pair. The problem is that the information is not sufficient to calculate the traffic matrix.

Three different research directions to estimate the traffic matrix from the link loads are introduced and compared in [Med02]. The problem formulation is as follows: Let Y be the vector of link measurements and $A \in \mathbb{R}^{L \times K}$ the matrix, where element $A_{lk} = 1$, if IE-pair k uses link l, and 0 otherwise. Let X be the traffic demand vector, in which element X_k denotes the traffic demand of IE-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 IE-pairs is usually much larger than the number of links. Three techniques for solving this under-defined linear problem listed in [Med02] are: 1) application of Linear Programming, 2) Bayesian Inference technique and 3) approach based on Expected Maximization (EM) algorithm to calculate the maximum likelihood estimates.

Chapter 4

Traffic engineering with MPLS

This chapter introduces first the MPLS architecture and its traffic engineering extensions. Then we review both static and adaptive load balancing algorithms proposed for the MPLS networks. Finally, we propose an adaptive load balancing algorithm and evaluate its performance under various networks and traffic conditions.

4.1 The architecture of MPLS

MPLS (Multi Protocol Label Switching) is a flexible technology that enables new services in IP networks and makes routing more effective. In MPLS two different approaches, datagram and virtual circuit routing, are combined to a compact technology. The idea of MPLS is to assign a label to each packet at the ingress node of the MPLS network. After that the packet is forwarded 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 Label Distribution Protocol (LDP). IETF defines three Label Distribution Protocols, LDP, CR-LDP and RSVP-TE. The architecture of MPLS is defined in [RFC3031].

4.1.1 The main elements

Label

A label used in MPLS is short and its length is fixed (e.g., Shim header, VPI/VCI header). A label is significant only locally and represents a particular FEC to which packet is designed to be forwarded. A 'labelled packet' is a packet with an encoded label. Actually, one packet can carry a number of labels, which are organized as a stack. A label stack can be used to support nested tunnels [Wan01a].

Label Switching Router

A Label Switching Router (LSR) is a node that is capable to forward native packets of the network layer and is aware of MPLS control protocols. LSRs can be divided into upstream and downstream LSRs with respect to a given FEC. The LSR that forwards packets to the direction of the dataflow is upstream and the LSR that receives packets is downstream.

Label Switching Path

A Label Switching Path (LSP) of a particular packet P is a sequence of routers $< R_1, ..., R_n >$, where R_1 is the 'LSP Ingress' and R_n is the 'LSP Egress' (see Figure 4.1). LSP Ingress pushes a label on some depth m of the packet's label stack. The routers between the ingress and egress nodes make forwarding decisions based on a label on level m. The egress node then forwards the packet using a label on the level $m - k$, where $k > 0$, or using some other forwarding procedure.

Next Hop Label Forwarding Entry

The Next Hop Label Forwarding Entry (NHLFE) is used when a labelled packet is transmitted forward. NHLFE includes information on the packet's next hop and the action to perform on the label stack of a packet. The label can be either replaced by a new label or the stack can be popped. Every label of an incoming packet can be associated to a particular NHLFE. In order to split traffic to different paths, there may be several NHLFEs for one label.

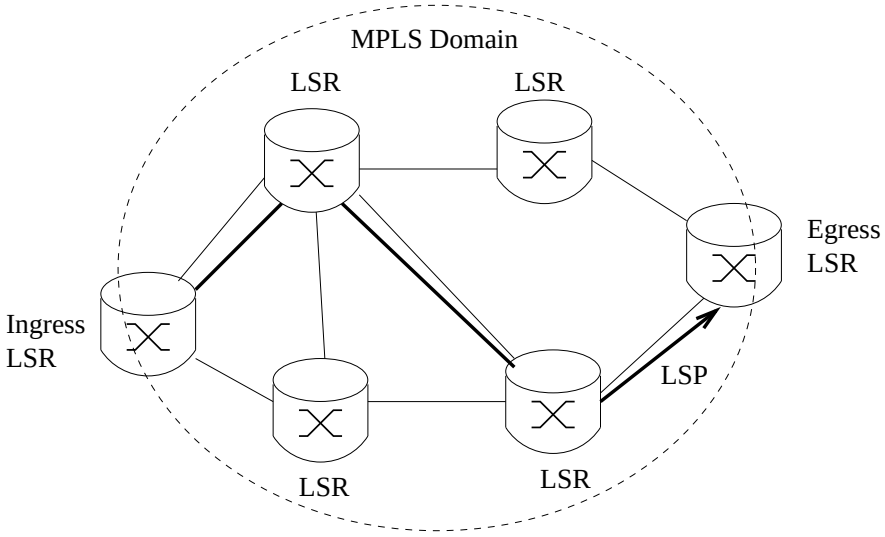

Figure 4.1: MPLS forwarding model

Incoming Label Map

Incoming Label Map (ILM) (also known as the label-switching table) describes the mapping between the label of an incoming packet and a set of NHLFEs. The LSR forwards the packet to the interface specified by the NHLFE. ILM can be compared with the forwarding table of a conventional IP router.

Label merging

Label merging refers to an LSR's capability to forward packets that have different source or different label to the single FEC (see Section 4.1.1) with a single label. The information that packets have, when they come from different interfaces with different labels, is lost in the label merging procedure.

Forwarding Equivalence Class

The information from some traditional routing protocol (e.g. OSPF) is preserved in the network layer routing in order to know how packets should be forwarded [Vis98] and in order to divide the forwarding space into Forwarding Equivalence Classes (FECs). FEC indicates which IP packets are forwarded over the same LSP and treated identically. The label of a particular packet assigns the packet to a certain FEC. In other words, each label is associated with a particular FEC. Packets are assigned to the particular FEC only once.

Forwarding granularity refers to the level of aggregation used in the forwarding

decision. MPLS supports many types of forwarding granularity. Packets can be classified into the same FEC according to the egress node, the IP destination address or the application flow. The classification according to the egress node represents the coarsest level of granularity and fits well in large networks, whereas the classification that depends on the application flow generates the finest granularity level, but is not well scalable.

Label Distribution Protocols

Label Distribution Protocol (LDP) is used in the path setup process. With LDP, one LSR notifies another LSR about the label/FEC bindings that it has made. These two LSRs are called "label distribution peers". The decision of binding a particular label to a particular FEC is done exclusively by the LSR that is downstream with respect to the binding. Distribution of label bindings can be done in two ways. In the first model an LSR requests from its neighbor LSR the binding for a particular FEC. In the second model, a downstream LSR distributes bindings straightforward without demand. LDP is also used to exchange information on MPLS capabilities between label distribution peers. MPLS allows multiple Label Distribution Protocols. Three protocols, LDP, CR-LDP and RSVP-TE, are standardized. LDP is defined in [RFC3036], CR-LDP in [RFC3212] and RSVP-TE in [RFC3209].

4.1.2 Traffic engineering with MPLS

One of the most significant applications of MPLS is traffic engineering, which is introduced in Chapter 2. In addition to explicit routing, MPLS provides mechanisms to route traffic between two edge routers along several paths. The capability to split traffic offers several advantages. Traffic can be routed successfully in the case of link failures using alternative paths, for example. However, the most important benefit of traffic splitting is the ability to balance the load. Load balancing reduces congestion and therefore improves the performance of the network.

The basic element of traffic engineering over MPLS is a traffic trunk that consists of the traffic that belongs to the same class and is routed along the same path. Traffic trunk attributes provide the ability to describe the characteristics of traffic trunks by the network operator. The assignment of traffic trunks through some network resources can be constrained by the network resource attributes. The requirements of traffic engineering over MPLS are described in [RFC2702].

MPLS can be developed to provide functionality of traffic engineering in large networks also. The costs are lower, because by MPLS it is possible to automate

traffic engineering functions. There are three basic problems in providing traffic engineering over MPLS. The first problem is how to map packets to FECs. The second problem is how to map FECs to particular traffic trunks, and the third problem concerns how to map traffic trunks onto the physical network through LSPs. The hierarchical presentation of different routing levels can be seen in Figure 4.2.

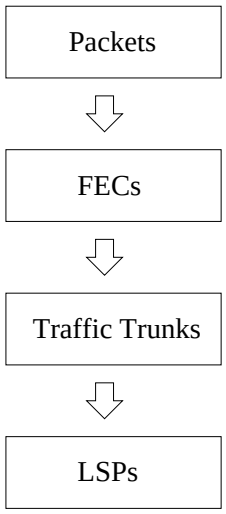

Figure 4.2: The hierarchy of MPLS streams

Induced MPLS graph

The concept of an induced MPLS graph is essential in traffic engineering over MPLS. An induced MPLS graph can be compared with a virtual topology of the overlay model. With the overlay model, a service provider establishes a virtual network that consists of all connections between the edge nodes. The set of routes is selected for the logical connections using constraint-based routing.

An induced MPLS graph is constructed from LSRs, which present nodes of the graph, and LSPs, which present the links of the graph that connect these LSRs logically. The abstract formulation of an induced MPLS graph according to [RFC2702] is as follows: Let $G = (V, E, c)$ be the capacitated graph of the physical topology of the network, where V is the set of nodes, E is the set of links, and c is the set of capacity and other constraints associated with V and E. Let $H = (U, F, d)$ denote the induced MPLS graph, where U is a subset of V so that nodes in set U present endpoints of at least one LSP, F is the set of LSPs in the graph, and parameter d presents demands and restrictions associated with F.

The induced MPLS graph is logically mapped onto the physical network and it can also contain many levels, which are hierarchically organized based on the concept of the label stack.

Traffic trunks

A traffic trunk is an aggregation of traffic flows that belong to the same class and are forwarded through a common path. A traffic trunk is an abstract presentation of traffic with specific characteristics. It is possible to aggregate traffic trunks or they can consist of many classes (referred to as the multi-class traffic aggregate). Traffic trunks encapsulate traffic between ingress LSR and egress LSR. They are unidirectional. There is a difference between a traffic trunk and an LSP through which the trunk traverses. In the operational context this means that traffic trunks can be moved from one LSP to another.

A bidirectional traffic trunk (BTT) contains two traffic trunks, the trunk from the origin node to the destination node and the trunk from the destination node to the origin node. The first trunk is referred to as the forward trunk, and the latter one as the backward trunk. Both traffic trunks are instantiated and destroyed together. Therefore neither of them can exist alone. The instantiation procedure must occur at one LSR or at a network element station through an atomic action (that is a sequence of indivisible statements). If the opposite traffic trunks are routed along the same physical path, BTT is said to be topologically symmetric, and if they are routed along different physical paths, BTT is said to be topologically asymmetric.

The operations of traffic trunks are essential when traffic engineering capabilities are considered. The basic operations are: establish, activate, deactivate, modify attributes, reroute and destroy. An instance of a traffic trunk is first created using the establish operation. The packets cannot be passed through the traffic trunk until the activate operation is done. When the route of a particular traffic trunk should be changed, the reroute operation is employed through an administrative action or using underlying protocols. Finally, the traffic trunks are removed using the destroy operation. At the same time the resources, like the label space and bandwidth, are released.

Traffic trunk attributes and resource attributes

The behaviours of traffic trunks are defined by the traffic trunk attributes. There are two ways to assign attributes. They can be assigned explicitly by an administration action or implicitly by the underlying protocols. The six basic attributes mentioned in [RFC2702] are listed and explained below. These attributes are essential when trying to achieve traffic engineering capabilities.

1. The traffic parameter attributes determine the characteristics (e.g. the peak rate and permissible burst size) of traffic streams to be routed along the traffic trunk.

2. The general path selection and management attributes define how paths are selected and maintained. The path selection can be done automatically using the underlying protocol or administratively by the network operator. When there are no restrictions associated with a particular traffic trunk, the paths can be selected using a topology driven protocol. If there exist some resource requirements, paths should be selected using constraint-based routing scheme.

3. The priority attribute determines how important a particular traffic trunk is in relation to the other trunks.

4. The preemption attribute defines preemption rules between traffic trunks. Four preempt rules for a traffic trunk are preemptor enabled, non-preemptor, preemptable and non-preemptable.

5. The resilience attribute defines how traffic trunks behave in fault situations. The basic problems to be solved by MPLS are fault detection, failure notification and recovery and service restoration.

6. The policing attribute defines how the underlying protocol responses to the situations where traffic trunks become non-compliant. The attributes indicate, if the traffic trunk should be rate limited, tagged or forwarded without policing action.

The routing of traffic trunks through some resources can be constrained using resource attributes that are part of the topology state parameters. The administratively configurable maximum allocation multiplier (MAM) defines the proportion of particular resources that is available for the traffic trunks. The resources can be under-allocated or over-allocated depending on the value of MAM. The resource class attribute determines which resources belong to the same class. The resource class concept is very useful when traffic engineering is actualized. For example, resource class attributes can be used to make uniform policies to a particular set of resources.

Constraint-Based Routing

Constraint-based routing, often referred to as QoS routing in the current literature, makes demand driven, resource reservation aware routing possible with the current topology driven IGP protocols. A constraint-based routing protocol computes automatically explicit routes for each traffic trunk at the ingress node of the trunk. The traffic trunk attributes, the resource attributes and additional topology state information can be used as an input for the routing process. The level of manual configuration and intervention can be reduced significantly using constraint-based routing.

The basic requirement of the constraint-based routing is the capability to automatically place traffic trunks onto paths that satisfy a set of constraints. The problem is demanding. However, there are many implementations of constraint-based routing in Frame Relay and ATM switches. It should not be difficult to extend implementations to cover requirements of MPLS. In the cases where routers use topology driven IGPs, the implementation of constraint-based routing can be done by extending current IGP protocols such as OSPF and IS-IS or by adding constraint-based routing process to those routers that can co-exist with current IGPs.

4.1.3 Multi-layer traffic engineering

MPLS provides means for traffic engineering in the Internet, specially from the link layer aspect. However, the flexibility provided by MPLS has to be extended to lower layers of the network, such as SDH/SONET and WDM, to satisfy the requirements set to the next generation networks [Iov03]. Generalized Multiprotocol Label Switching (GMPLS), also referred to as Multiprotocol Lambda Switching (MPλS), provides a standardized control plane for open and interoperable optical networks [Ban01]. The main enhancements include a new Link Management Protocol (LMP), capability of OSPF/IS-IS protocols to advertise availability of optical resources, ability to explicitly define LSPs across the optical core by RSVP, and scalability enhancements such as hierarchical LSP formation.

The idea of multi-layer traffic engineering relies on the co-operation of the link layer and optical layer protocols. An example of a reference scenario of multi-layer TE can be found in [Iov03] and is depicted in Figure 4.3. At the MPLS layer, the LSRs are connected by LSPs via MPLS links, whereas at the optical layer optical cross-connects (OXC) are connected by lightpaths via optical links. One task of multi-layer traffic engineering is to determine how these LSPs and lightpaths can be implemented in a cost effective way in the whole network. Also recovery and restoration from the link failures of the optical network require attention in the multi-layer approach.

4.2 Solving the static load balancing problem in MPLS networks

MPLS provides capabilities to predefine the paths used between each source and destination node, which is also called as explicit routing. MPLS also enables splitting traffic to several paths instead of one single path used by traditional routing protocols. Traffic splitting gives the ability to balance load and thus

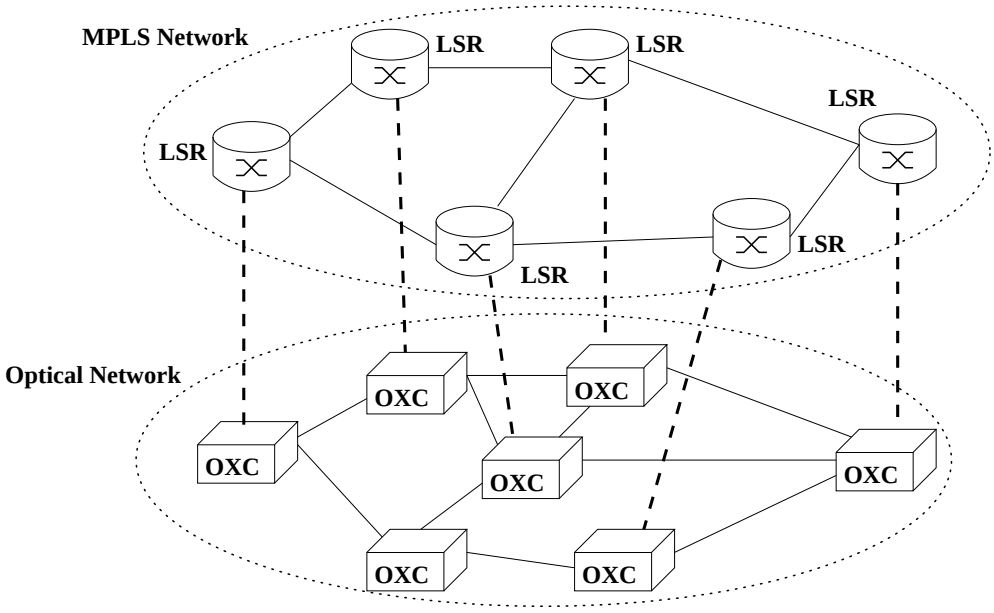

Figure 4.3: Multi-layer reference scenario

improve the performance of the network. There are several algorithms that try to solve the static load balancing problem presented in Section 2.3.1 by taking advantage of the special capabilities of MPLS networks. Optimization of the problem is based on off-line computation and a basic assumption is that the traffic demand matrix can be estimated somehow.

4.2.1 Static load balancing algorithms

Ott et al. [Ott01] consider a 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 to obtain the traffic allocation whose performance is close to that of optimal routing. The approximative algorithm solves first the traffic allocation by LP-optimization, where optimization objective is minimization of the maximum link utilization (compare with problem (2.8)). From this traffic allocation the corresponding paths are inferred and, after that, the traffic is reallocated to those paths using NLP-optimization (such as optimization problem (2.2) with objective function (2.4)). This approach allows the separation of computation into two levels and at the two time scales: the paths are calculated infrequently off-line, whereas the traffic allocation to those paths is calculated more frequently in a distributed manner.

The granularity aspects of traffic engineering are studied in [Sri00]. The traffic granularity refers to the level of traffic aggregation and the time granularity refers to the variations of traffic demands as a function of the length of measurement

time. Simple heuristics for load balancing using a particular level of traffic granularity is also developed. In this algorithm, depending on the level of granularity, traffic demands from an ingress node to an egress node are divided into g streams (e.g. using some hashing function) and each stream is routed sequentially one at a time to the shortest path defined by Dijkstra's algorithm using the mean delay of an $M/M/1$-queue as the link cost. The result of the paper is that a small change in the traffic granularity can improve the performance of the network significantly, but this is not necessarily true for finer time scales since the greater variability in traffic offsets the achieved improvements in the performance.

4.2.2 Comparison of the algorithms

In [Sus02] we have compared how the load balancing algorithms explained in the previous section approximate the optimal load distribution using off-line computation in the static traffic scenario. The objective is the minimization of the mean delay of the network. As an example, the mean delay as a function of the traffic load in the test-network of 10 nodes (will be specified in the next section) is shown in Figure 4.4.

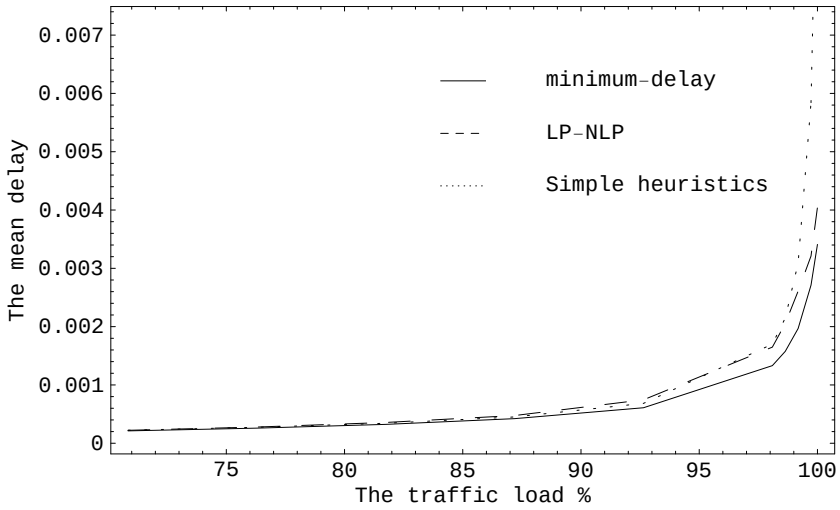

Figure 4.4: The mean delay as a function of traffic load. Granularity parameter in the simple heuristics is $g = 128$.

From the figure we can see that both approximations (referred to as "LP-NLP" and "Simple heuristics") work well until the network load is close to 100 %. However, the computation time of LP-NLP approximation is 10 times shorter than that of the direct optimization of the minimum delay.

4.3 Adaptive approach for balancing the load

The static load balancing problem of Section 2.3.1 can be formulated and solved only if we have precise information on all the traffic demands d_k. It may well be the case that such information is either imprecise, outdated, or totally missing. In such a case, another approach is needed. Adaptive algorithms reviewed in this section utilize the special properties of MPLS when trying to improve the performance of the network.

4.3.1 Review of adaptive load balancing algorithms

Dinan et al. [Din00] study analytically how the traffic can be mapped to multiple parallel paths in an optimal way. In the model 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. The assumption of the algorithm is that the network state can be defined by summing the traffic loads on the LSPs of each IE-pair.

Elwalid et al. [Elw01] 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 the packet loss are measured periodically by sending probe-packets between the ingress and the egress node of a Label Switched Path. When a change of predefined size in these traffic conditions is perceived, the algorithm moves from the monitoring phase to the load balancing phase. In this phase the algorithm tries to equalize congestion measures among the LSPs by approximating the gradient-projection algorithm (introduced in Section 2.3.4). Since the required first derivative lengths in the algorithm are not directly available, they are estimated by averaging several measurements. After the load balancing phase the algorithm moves back to the monitoring phase.

In paper [Güv04] some criticism concerning 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 [Güv04] proposes some enhancements to MATE. Indeed, the underlying infrastructure is not necessarily MPLS, but an overlay architecture is assumed that establishes multiple paths between source and destination pairs. The algorithm of [Güv04] is based on calculation of gradient vector by simultaneous perturbation stochastic approximation (SPSA), which provides a convergence close to optimal value with relatively small number of measurements, even if they are noisy.

Song et al. [Son03] 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. The LSP for the incoming traffic is selected based on congestion and the length of the path. When a particular link becomes congested, those ingress and egress pairs that use a long path have to move their traffic away from that link. A new LSP is selected randomly according to probabilities that depend on the path utilizations and the path lengths.

A mixed integer programming (MIP) problem for hop-count and path-count constrained multipath routing is presented in [Seo01]. 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 calculates 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.

4.4 Proposal for adaptive load balancing by MPLS

We find some drawbacks in the algorithms introduced in Section 4.3.1. First, in [Elw01] traffic engineering is based on the measured traffic conditions between each ingress and egress pair. The use of active measurements load the network, and moreover, these measures offer overlapping 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 [Son03] 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.

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 edge routers independently of each other. In addition, a numerical method is developed to evaluate the performance of the load balancing algorithm. The method is applied to three test networks.

4.4.1 Dynamic load balancing problem

In this section we consider the dynamic load balancing problem with incomplete information. More precisely, we assume that the traffic demands are unknown but the link loads are periodically measured using a monitoring system as in [But03]. As noted in the previous chapter, one cannot derive the traffic demands from the link loads. Therefore, the problem cannot be formulated, let alone solved, as a static load balancing problem.

As before, consider a network consisting of nodes $n \in \mathcal{N}$ and links $l \in \mathcal{L}$. The network is loaded by the traffic demands between IE-pairs $k \in \mathcal{K}$, but these demands are unknown. Instead, we assume that the link loads are measured periodically at times t_i. These measured link loads $\hat{y}_l(i)$ should be understood as time-averages over the whole measurement period (t_{i-1}, t_i), and not as instantaneous values. On the other hand, due to traffic fluctuations in different time scales, the measured loads deviate from the long-term time averages in a random manner (we will later assess the effect of the traffic fluctuations by numerical experiments).

We assume that the information on the measured loads is distributed to all edge routers in the MPLS network. This can be done in a similar way as the link states are distributed to all routers within an AS in OSPF [RFC2328]. 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).

Each IE-pair k has a predefined set $\mathcal{P}_k$ of LSP's. Let $\phi_p(i)$ denote the *proportion* of traffic allocated to path p at time t_i. These splitting ratios are the control variables in our dynamic load balancing problem satisfying, for all k and i,

$$\sum_{p \in \mathcal{P}_k} \phi_p(i) = 1.$$

Splitting ratios $\phi_p(i)$ are gradually adjusted based on the measured link loads $\hat{y}_l(i)$. Thus, the algorithms to determine these ratios may be called *adaptive*.

It is important to note that, since these splitting ratios determine the proportions of traffic, and not the absolute amounts of traffic, allocated to LSP's, there is no need to know the traffic demands. 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 [Son03], 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_{i-1}, t_i), the splitting ratios $\phi(i) = (\phi_p(i); p \in \mathcal{P})$ in such a way that they converge, as soon as possible, to the unknown optimal values ϕ_p^* of the corresponding static load balancing problem, e.g. (2.2),

$$\phi_p^* = \frac{x_p^*}{d_k}, \qquad \text{for all } p \in \mathcal{P}_k.$$

4.4.2 An adaptive and distributed load balancing algorithm

In this section we describe an adaptive and distributed algorithm to solve the dynamic load balancing problem presented in the previous section. The algorithm is heuristic, and thus suboptimal. However, it seems to work quite well, as we will see in the following section, where the performance of the proposed algorithm is evaluated numerically.

First we note that, since the measured loads $\hat{y}_l(i)$ are distributed to all edge routers, the decisions concerning the splitting ratios $\phi_p(i)$ can be made in a distributed way. Thus, in our *distributed* algorithm, each edge router independently determines the splitting ratios $\phi_p(i)$ for all those paths p for which it is the ingress node.

Next we describe how an edge router determines the splitting ratios for any of the IE-pairs for which it is the ingress node. Consider one such IE-pair, say k. See Figure 4.5 for an illustration.

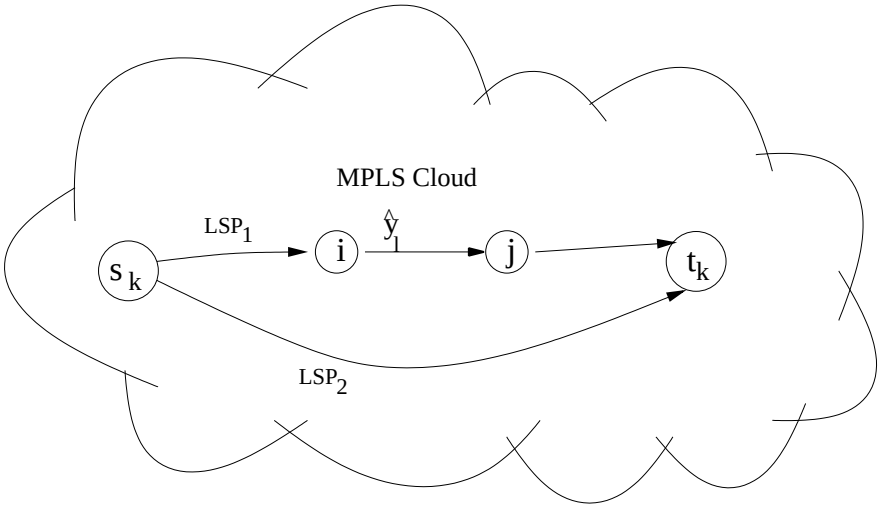

Figure 4.5: The network model

As our algorithm is *adaptive*, the splitting ratios will depend on the measured loads $\hat{y}_l(i)$. More precisely, they will depend on the estimated path costs $D_p(\hat{y}(i))$,

where $\hat{y}(i) = (\hat{y}_l(i); l \in \mathcal{L})$ denotes the vector of measured loads. The definition of the path cost $D_p(\hat{y}(i))$ depends on the formulation of the original optimization problem. If the purpose is to minimize the total cost (2.2), such as the mean delay, then we define two different path costs. One is the total cost along the path,

$$D_p(\hat{y}(i)) = \sum_{l \in p} C_l(\hat{y}_l(i)), \tag{4.1}$$

and the other one is its first derivative length,

$$D_p(\hat{y}(i)) = \sum_{l \in p} C_l'(\hat{y}_l(i)), \tag{4.2}$$

But if the objective is to minimize the maximum cost (2.5), such as the maximum utilization, then the path costs are naturally defined by

$$D_p(\hat{y}(i)) = \max_{l \in p} C_l(\hat{y}_l(i)) \tag{4.3}$$

The idea is simply to alleviate the congestion on the most costly path (among the paths belonging to the same $\mathcal{P}_k$) 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 $\mathcal{P}_k$. 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. As discussed in Section 2.4, it is well known that feedback control systems are prone to instability if the gain in the loop is too large. Thus, to avoid harmful oscillations, we let the splitting ratios change only with minor steps. The step size is determined by the granularity parameter g that will be defined below. A finer granularity is achieved by increasing the value of g. We will study an appropriate choice of g by numerical experiments in Section 4.4.4.

Algorithm for determining the splitting ratios. At time t_i, after receiving the information concerning all the measured loads $\hat{y}_l(i)$, the edge router s_k operates as follows:

1. Calculate the path costs $D_p(\hat{y}(i))$ for each path $p \in \mathcal{P}_k$.
2. Find the path $q \in \mathcal{P}_k$ with maximum cost, i.e. $D_q(\hat{y}(i)) = \max_{p \in \mathcal{P}_k} D_p(\hat{y}(i))$, and decrease its splitting ratio as follows:

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

3. Choose another path $r \in \mathcal{P}_k$ either

a) randomly, or

b) so that the path cost is minimized, i.e. $D_r(\hat{y}(i)) = \min_{p \in \mathcal{P}_k} D_p(\hat{y}(i))$,

and increase its splitting ratio as follows:

$$\phi_r(i) = \phi_r(i-1) + \frac{1}{g}\phi_q(i-1).$$

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

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

4.4.3 Numerical evaluation of the algorithm

Next we evaluate the performance of the proposed algorithm. First, a simple but efficient numerical evaluation method is developed. Thereafter, in Subsection 4.4.4, the results of applying this evaluation method to three different test networks are presented.

The evaluation method is iterative and runs as follows. The test network includes the nodes n, links l, IE-pairs k, paths p and the traffic demands d_k.[1] We add a random component to the traffic demands, which describes how the average traffic rate in a measurement period, $\tilde{d}_k(i)$, fluctuates around the deterministic demand, d_k, resulting in

$$\tilde{d}_k(i) = d_k + \epsilon_k(i), \quad \text{for all } k \in \mathcal{K}, \tag{4.4}$$

where the $\epsilon_k(i)$ are selected as independent Gaussian random variables with mean 0 and variance $\delta^2 d_k^2$. Cao et al. use a corresponding traffic model in [Cao00], for example. Splitting ratios ϕ_p are initiated by allocating the traffic demands to the paths with the minimum hop-count. If multiple shortest paths exist, the path is selected randomly.

At each iteration i, the measured link loads $\hat{y}_l(i)$ are determined from the splitting ratios $\phi_p(i-1)$ and traffic demands $\tilde{d}_k(i)$ as follows:

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

The new splitting ratios $\phi_p(i)$ are determined from these measured loads $\hat{y}_l(i)$ as presented in the previous section.

By this way we are able to numerically determine

[1] Note that the traffic demands are used only for the *evaluation* of the proposed load balancing algorithm. The algorithm itself does *not* use any information on these demands.

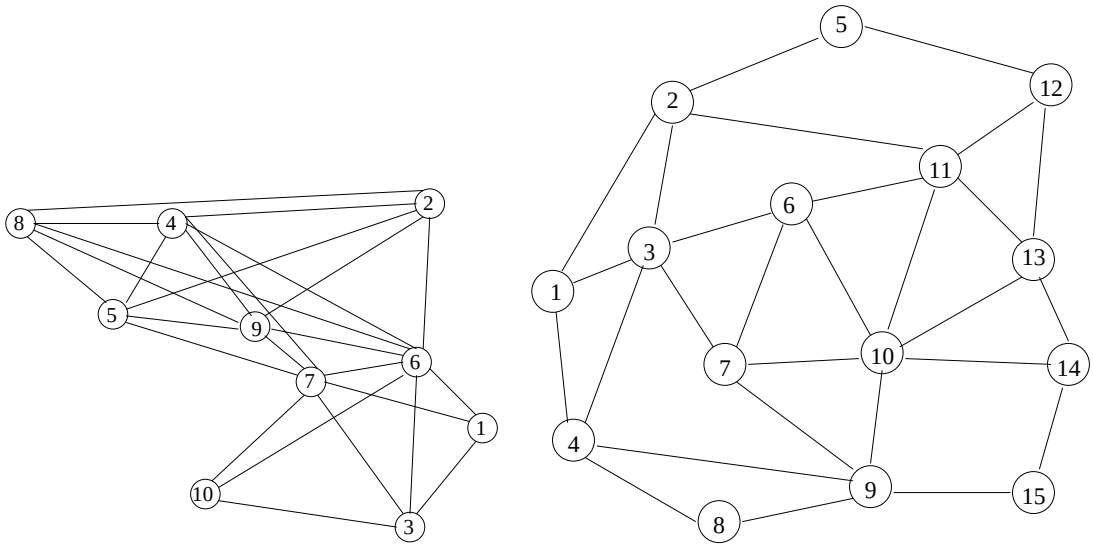

Figure 4.6: Left-hand-side: 10-node network. Right-hand-side: 15-node network.

(i) whether the cost value produced by the algorithm converges or not,

(ii) whether the convergence happens without oscillations,

(iii) how fast the convergence is (i.e. in how many iterations it takes), and

(iv) how much the limiting cost value differs from the optimal cost.

4.4.4 Numerical results

Three different test networks were used with the following characteristics (see Figures 4.6 and 4.7):

1. 10 nodes, 52 links, and 72 IE-pairs;
2. 15 nodes, 56 links, and 4 IE-pairs;
3. 20 nodes, 102 links, and 380 IE-pairs.

The first and third networks are random networks generated to test MPLS-OMP algorithm by C. Villamizar [Vil99a]. The second one was taken from [Kod00]. The number of possible paths, which may be huge, was limited by requiring that the paths $p \in \mathcal{P}_k$ do not have any common links. This policy results in a reasonable number of paths and provides the greatest diversity of links. Without the limitations to the number of the paths, the algorithm is hardly scalable since the overhead produced by LSP set-up is huge.

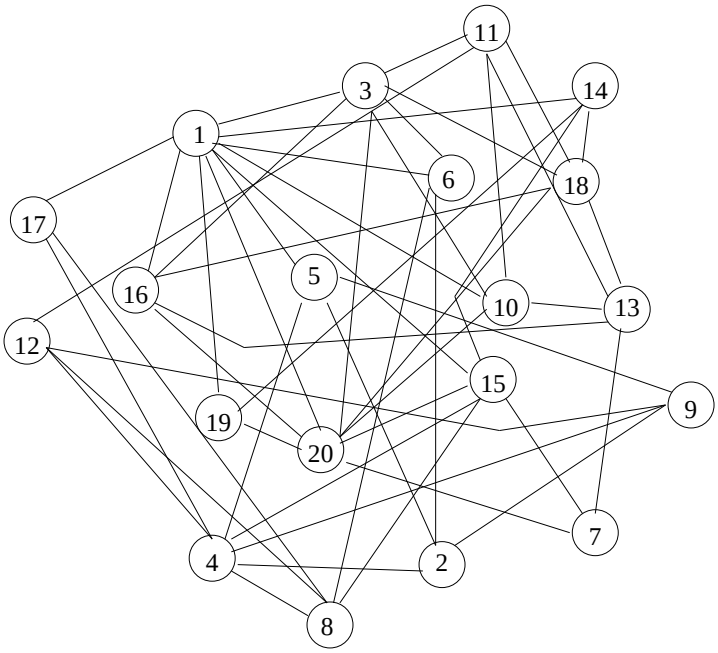

Figure 4.7: 20-node network.

Minimization of the maximum link utilization

First we present the results related to the minimization of the maximum link utilization (2.5) with cost function (2.6). The effect of traffic fluctuations on measured link loads is ignored, that is, variation parameter $\delta = 0$. The following variants of our adaptive load balancing algorithm were studied:

(i) `RNDPTH`: path cost (4.3) and rule 3a in the algorithm.

(ii) `MINCST`: path cost (4.3) and rule 3b in the algorithm.

In the following figures we depict the maximum link utilization as a function of the number of iterations i for different levels of granularity g. The results of the adaptive algorithm are compared with the optimal value (lower bound) and with the maximum link utilization resulting in the minimum hop-count routing (upper bound). The optimal value is obtained from the static load balancing problem (2.5) by writing a corresponding LP-formulation (2.9) and solving the problem using Minos 5 solver module. The maximum link utilization resulting in the minimum hop-count routing is taken from the first iteration round of the algorithm.

We consider first the network with 10 nodes. Figure 4.8 shows the results related to variant `RNDPTH` and Figure 4.9 the results related to variant `MINCST` of the algorithm. 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. The values

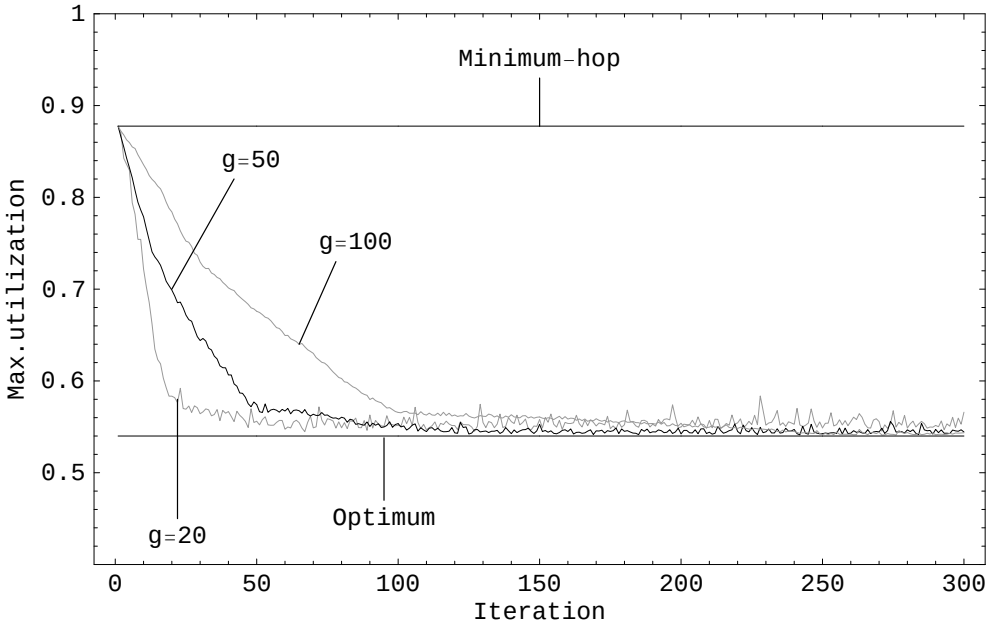

Figure 4.8: The maximum link utilization in the 10-node network as a function of the number of iterations for different levels of granularity g. Variant `RNDPTH`.

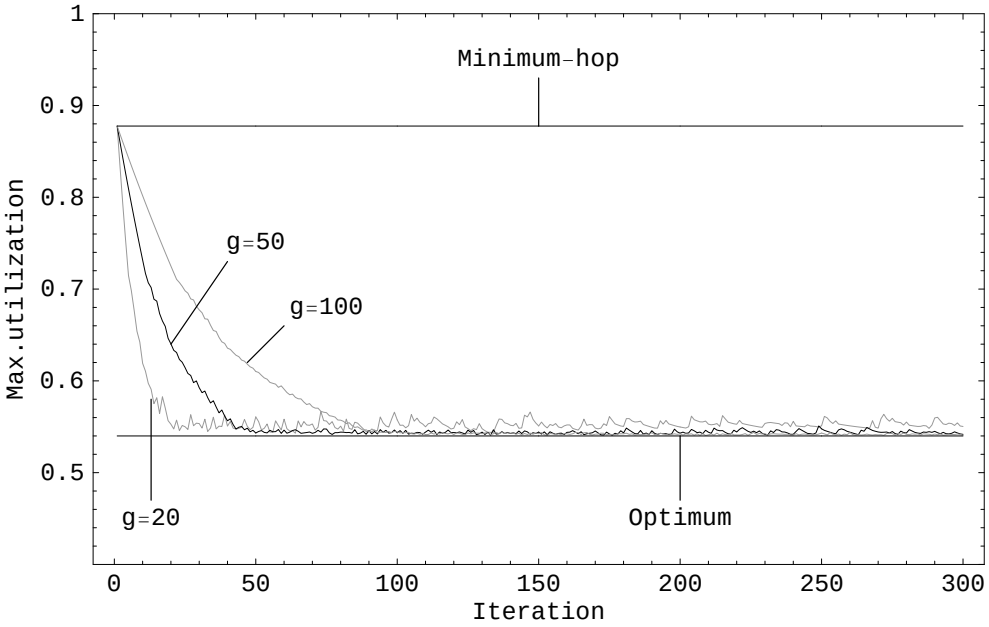

Figure 4.9: The maximum link utilization in the 10-node network as a function of the number of iterations for different levels of granularity g. Variant `MINCST`.

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 oscillations are outstanding. With finer granularities ($g = 50, 100$), oscillations disappear almost completely. The two variants of the adaptive load balancing algorithm do not differ significantly from each other in this scenario.

The results related to the `RNDPTH` variant applied to the test networks with 15 and 20 nodes are shown in Figures 4.10 and 4.11, respectively. The convergence times are only two times as large as in the 10-node network in spite of the huge growth in the complexity of the network. In the case of a 20-node network there remains a gap between the steady state and the optimal values. However, the improvement to the performance of the minimum hop-count routing is outstanding.

Minimization of the total mean delay

Now we present the results related to the minimization of the total mean delay with cost function (2.4), when the traffic fluctuations are ignored. The following variants of our adaptive load balancing algorithm were studied:

(i) `RNDPTH`: path cost (4.1) and rule 3a in the algorithm.

(ii) `MINCST`: path cost (4.1) and rule 3b in the algorithm.

(iii) `MINDRV`: path cost (4.2) and rule 3b in the algorithm.

For comparison, we also studied the performance of the gradient-projection (`G-P`) algorithm for the multi-commodity flow problem as presented in Section 2.3.4. The value to which the G-P-algorithm converges we can see also the optimum of the delay optimization problem. The total mean delay resulting in the minimum hop-count routing is calculated from the link load values after the first iteration round of the algorithm.

Figure 4.12 shows the results related to the 10-node test network with a coarse granularity, $g = 10$. The traffic demands are such that the min-hop routing results in the mean delay of 0.000179 while the optimal value is 0.000122. The convergence of the algorithm is achieved approximately in 20 iterations in all cases. We note that 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 the variant `MINDRV` seems to converge closer to optimal value than the other variants, whereas the variant `RNDPTH` converges furthest.

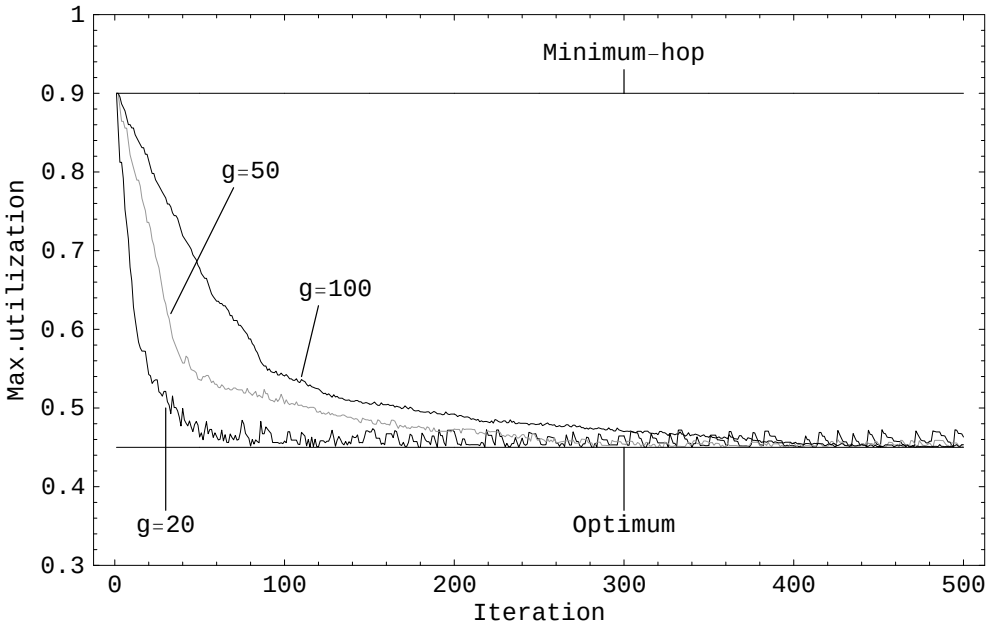

Figure 4.10: The maximum link utilization of variant RNDPTH as a function of the number of iterations for different levels of granularity g. 15-node network.

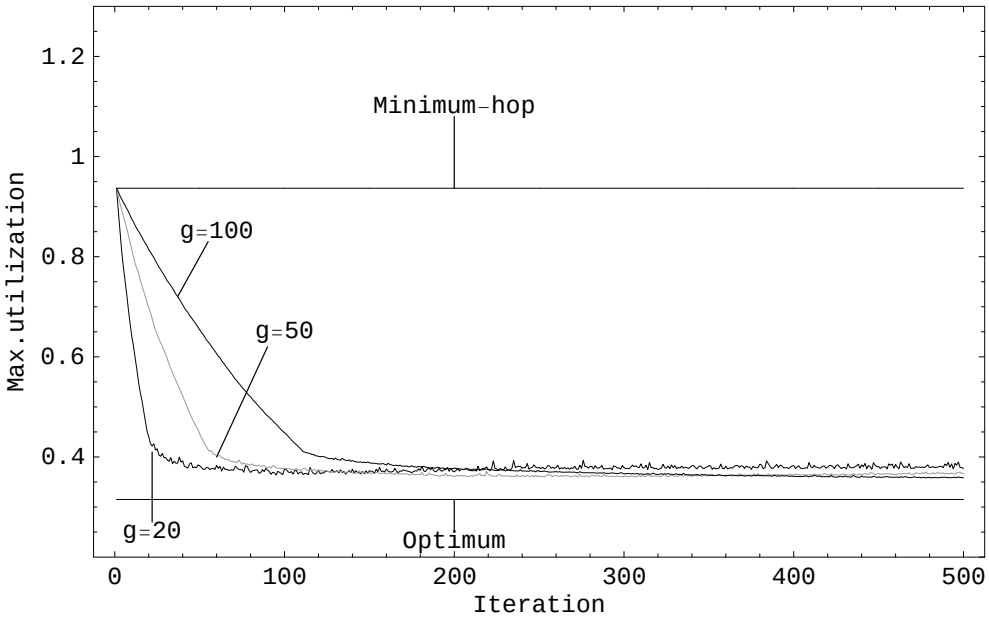

Figure 4.11: The maximum link utilization of variant RNDPTH as a function of the number of iterations for different levels of granularity g. 20-node network.

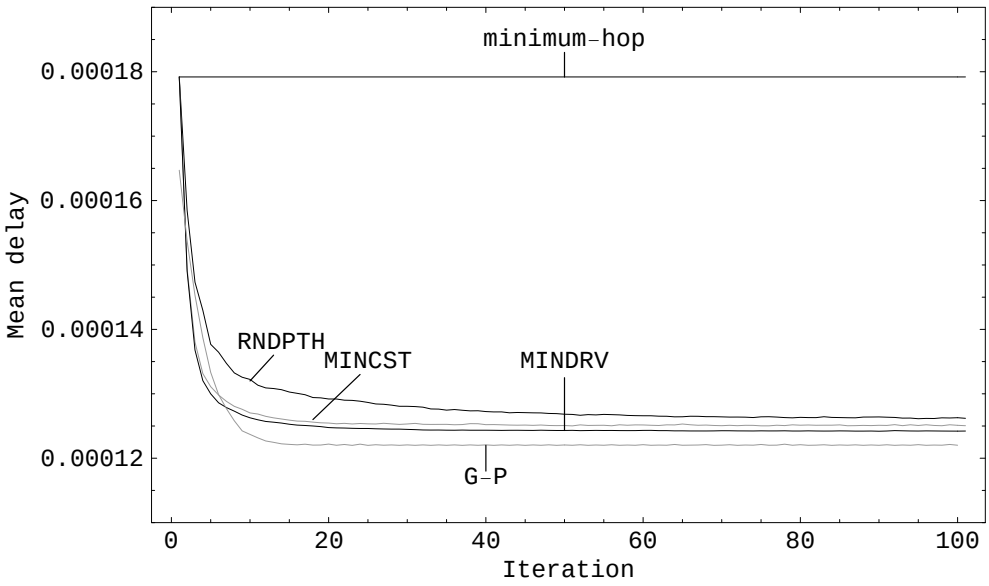

Figure 4.12: The total mean delay in the 10-node network as a function of the number of iterations for the variants `RNDPTH`, `MINCST`, and `MINDRV` of the adaptive load balancing algorithm with granularity $g = 10$ together with the `G-P` algorithm.

In Figures 4.13 and 4.14 we show the corresponding results of the minimization of mean delay in 15-node and 20-node networks, respectively. The order of the variants is the same as in the 10-node network, except for the variant `MINCST` which works extremely well in the 15-node network. The number of iterations required to convergence are very similar in all three networks.

In comparing the results of minimization of the mean delay to the min-max problem of the previous subsection, we note that oscillations are much smaller in the first case. Thus we are able to increase the step-size and achieve faster convergence. The mean delay as the objective function is more stable as the min-max objective.

Effect of traffic fluctuations on the link load estimation

To study the effect of random traffic fluctuations on measured link loads we add a random component to the traffic demands. In the previous subsections the variance parameter of equation (4.4) has been $\delta = 0$, but in this subsection we model random fluctuations by changing δ.

One common approach to reduce the variability of measurements is to average the measurements over a time period. In the exponential weighted moving av-

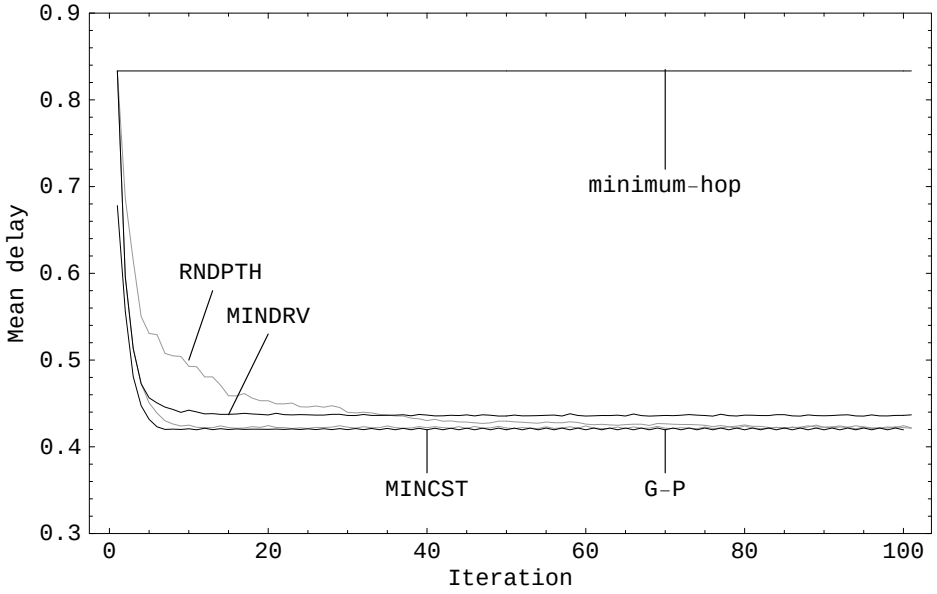

Figure 4.13: The total mean delay in the 15-node network as a function of the number of iterations for the variants RNDPTH, MINCST, and MINDRV of the adaptive load balancing algorithm with granularity $g = 10$ together with the G-P algorithm.

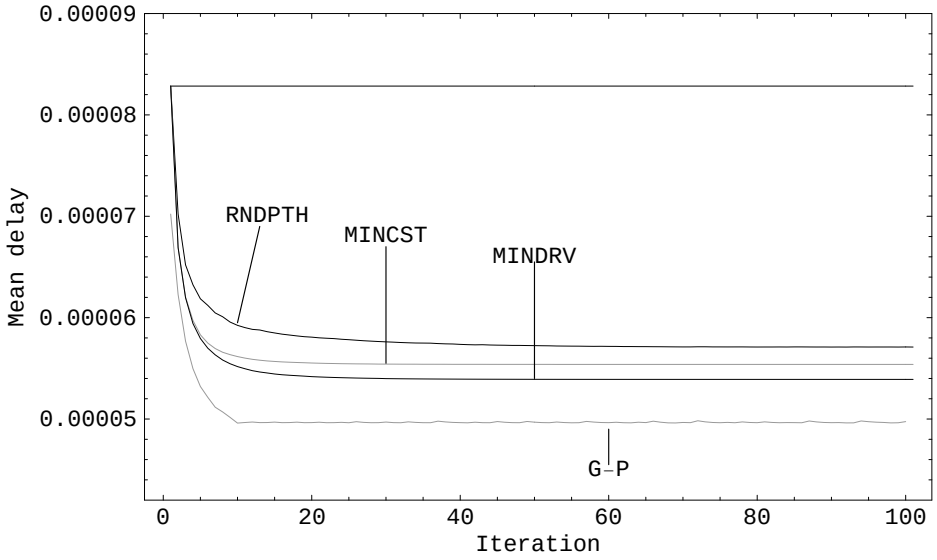

Figure 4.14: The total mean delay in the 20-node network as a function of the number of iterations for the variants RNDPTH, MINCST, and MINDRV of the adaptive load balancing algorithm with granularity $g = 10$ together with the G-P algorithm.

erage (EWMA) method the short-term measurements $\hat{y}_l(i)$ are replaced by their averaged values

$$\bar{y}_l(i) = \alpha\hat{y}_l(i) + (1-\alpha)\bar{y}_l(i-1),$$

where $\alpha \in (0, 1]$ controls the time span of the EWMA method. A smaller value of α gives a longer time span. The highest value $\alpha = 1$ corresponds to the original adaptive load balancing algorithm (without any averaging of the measurements).

We studied the effect of traffic fluctuations together with the compensating EWMA method on the minimization of the maximum link utilization problem. Figure 4.15 shows the results related to the 10-node test network. The left upper corner corresponds to the original `RNDPTH` variant of our adaptive load balancing algorithm ($\alpha = 1$). In the other parts of the figure we combined the `RNDPTH` variant with the EWMA method with different values of α. Figure 4.16 gives corresponding results for the `MINCST` variant. The coefficient of variation parameter is $\delta = 0.3$ in all the evaluations.

We find that the random traffic fluctuations induce oscillations to the maximum link utilization. However, the use of EWMA is not necessary since we may get rid of oscillations by increasing g. The smaller shifts in the traffic allocation average the measurements and thus the algorithm does not react too aggressively to unreliable information of the traffic conditions.

Effect of traffic demand changes

In addition to random traffic fluctuations at the time scale of the measurement period, we studied the effect of long-term variations in the traffic matrix itself. These can be thought of either as ordinary variations in traffic demand during a day or more unpredictable variations due to some external events. In this scenario the traffic demands change drastically three times during the evaluation period (after 100, 200 and 300 iterations, respectively) according to the same distribution as in the previous subsection. We have studied how the adaptive algorithm works as the objective is to minimizes the maximum link utilization.

The results for the variant `RNDPTH` in the 10-node network is shown in Figure 4.17. As the traffic demand changes, there is a spike in the performance of the network. However, the traffic load is balanced soon again. Corresponding results for the variant `MINCST` are presented in Figure 4.18. There are not remarkable differences between the variants.

As a summary of these experiments we find that our adaptive algorithm performs well also under long-term traffic variations reacting approximately as fast as the steady state was reached in our ordinary experiments.

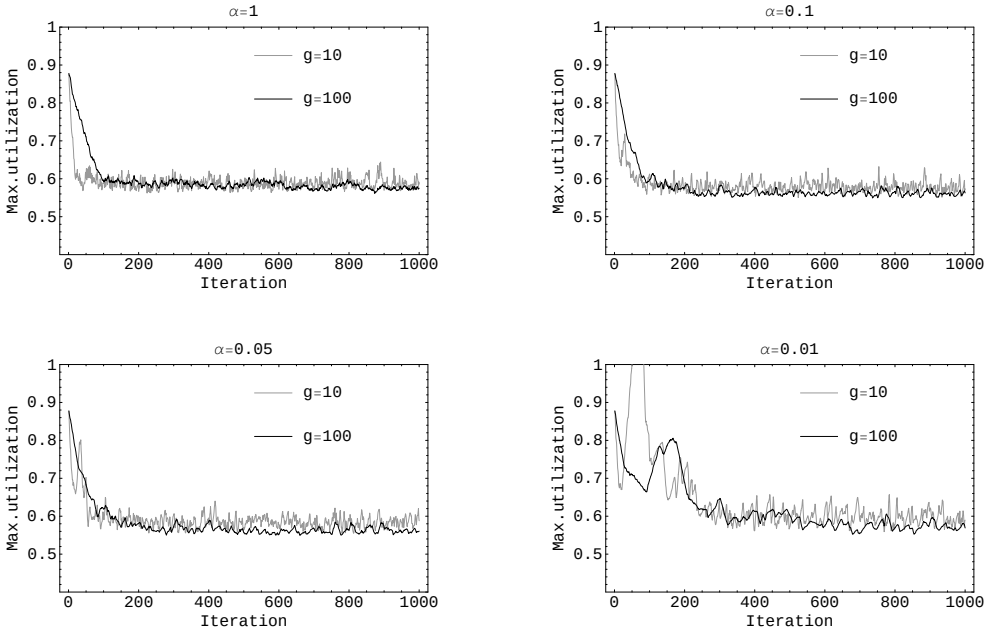

Figure 4.15: The maximum link utilization of variant RNDPTH combined with EWMA in the 10-node network as a function of the number of iterations.

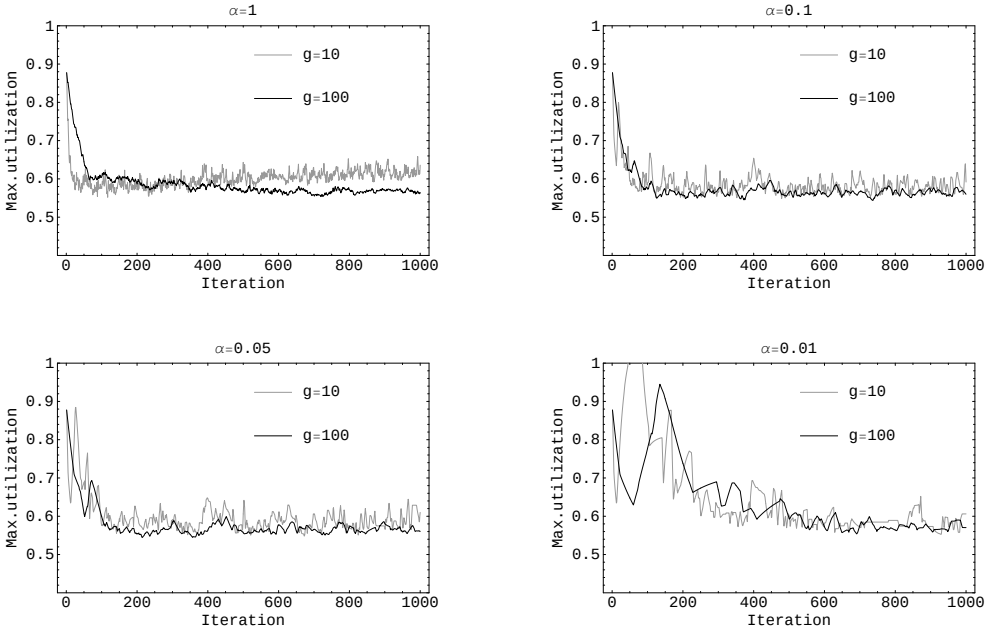

Figure 4.16: The maximum link utilization of variant MINCST combined with EWMA in the 10-node network as a function of the number of iterations.

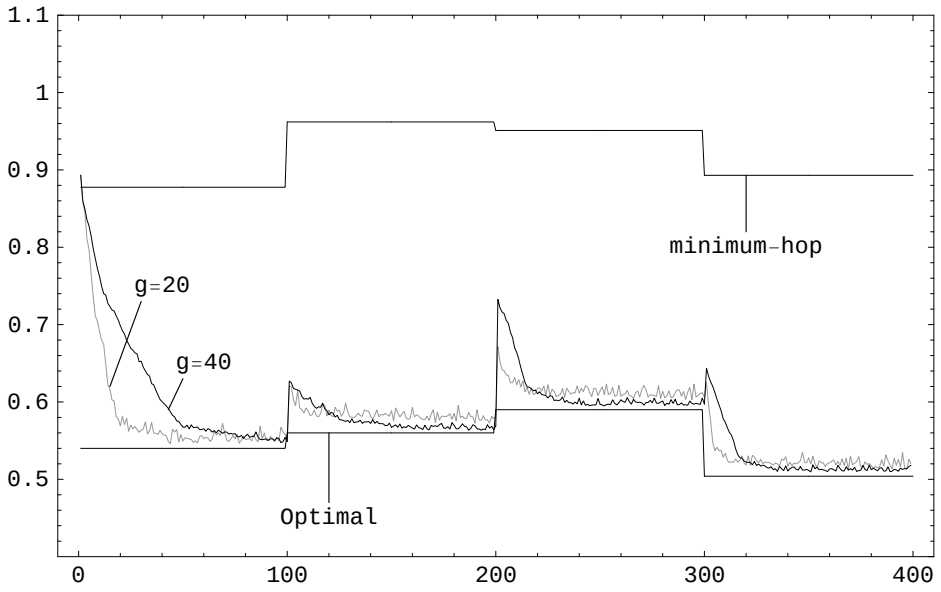

Figure 4.17: The maximum link utilization of variant RNDPTH in the 10-node network as a function of the number of iterations.

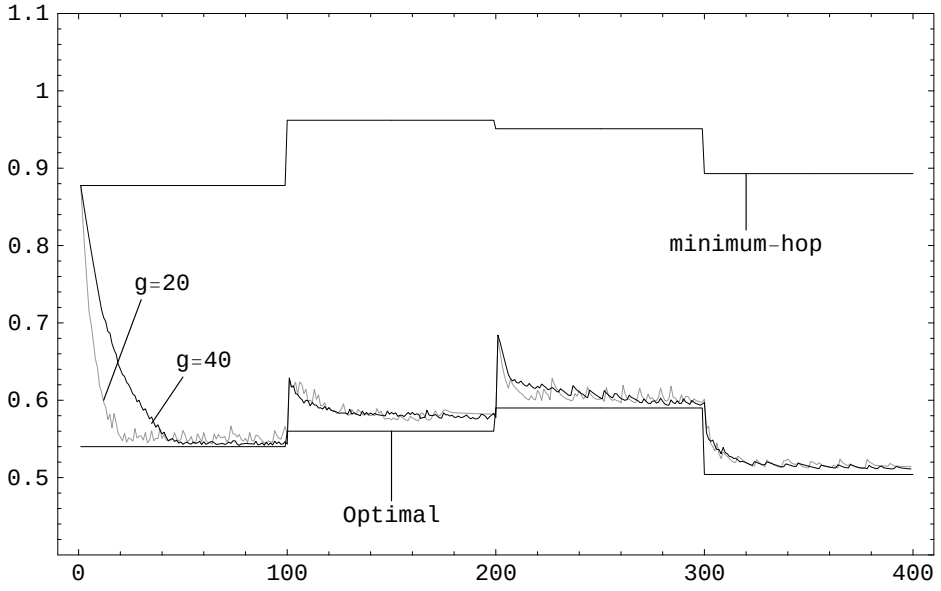

Figure 4.18: The maximum link utilization of variant MINCST in the 10-node network as a function of the number of iterations.

4.5 Conclusion on TE in MPLS networks

In this chapter we first described the architecture of MPLS and then reviewed both static and adaptive traffic engineering methods in an MPLS network. Then we developed our own adaptive and distributed algorithm for load balancing using a simple heuristic approach. The idea is to measure the link loads periodically and then gradually move traffic from the congested part of the network to the less congested part. We considered two objectives, the minimization of the maximum link utilization or the total mean delay.

The obtained steady state results were in many cases very close to the optimum and in all cases much better than those of the standard min-hop routing. When the maximum link utilization was minimized, the variants `RNDPTH` and `MINCST` gave quite similar results. The main difference is that the convergence times were longer in the variant `RNDPTH`. On the other hand, the latter variant, which allows only traffic shifts that improve the local performance, led to sub-optimal results in some traffic scenarios.

The mean delay of the network is minimized most effectively by the variant `MINDRV` of the algorithm, which moves traffic from the path with the maximum first derivative length to the path with the minimum first derivative length. The convergence times to obtain the steady state were reasonable in all cases, even when the complexity of the network was substantially increased.

In general, the algorithm works well when granularity parameter is $g > 20$, which means that approximately 5% of the traffic load is moved at a time. When the mean delay is minimized, the step-size can be greater due to the stability of the results.

Chapter 5

Traffic engineering with OSPF

The aim of this chapter is to study traffic engineering in OSPF networks. As in the previous chapter, first we introduce the underlying infrastructure and then review the main achievements in the area of TE with OSPF. We study how the load can be balanced by applying the approach developed in the previous section to OSPF networks.

5.1 Conventional routing in the Internet

In IP routing, the routers make independent decisions about how to forward packets to destinations. When a packet arrives, the router forwards it according to its destination IP 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 the network changes [Cha00].

Dynamic routing protocols can be divided into interior routing protocols and exterior routing protocols. The most common interior routing protocols are 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 and calculate the best paths to all the destinations in the network.

RIP is applicable for small networks. With RIP, the cost information about the paths to the destinations is periodically changed between neighboring routers. However, there are problems with RIP due to the use of a single metric and a limited hop distance.

In a large network OSPF is recommended instead of RIP. The advantages of OSPF are hop-by-hop routing, fast dissemination of routing information and hierarchical routing [Zee99]. After changes in the network, the new paths are updated quickly producing a minimal amount of overhead. The shortest paths are calculated using the Dijkstra algorithm.

Border Gateway Protocol (BGP) is a widely used exterior routing protocol in the Internet. BGP is a path-vector protocol and it is responsible for exchanging information between Autonomous Systems (ASs) by disseminating accessibility information.

5.2 OSPF routing

The idea of OSPF, developed by the IETF, is that each router computes the shortest paths to the destinations based on the map of the network, which is updated every time when changes in network topology are detected. This map is represented by a data structure called the *link state database*. In the OSPF routing it is essential that the databases of all routers are identical and thus routing loops are avoided. The documentation of OSPF version 2 is presented in [RFC2328]. A good overview of the subject is also provided in the book [Hui95].

5.2.1 Link state database

The routing decisions in the OSPF are based on the link state data base, which includes information about the state of the links in the network. The link state database is constructed by collecting the link state advertisements (LSA) sent by the other routers. Each advertisement consists of the link state header and the content field, which includes metrics used in computation of the shortest paths. There are different types of LSAs in OSPF; 1) Router-LSA describes which links start from the router, 2) network-LSA provides information about the network interfaces, 3) summary-LSAs describe different destinations, such as an IP network or AS border router. Finally, 4) external-LSAs inform about the routes to destinations in external ASs.

Since traffic from applications have different characteristics, also the best paths vary from one application to another. Thus it is convenient to have multiple link metrics in the process of computing shortest paths. Indeed, OSPF version 2 supports multiple metrics in parallel by adding variable number of TOS fields to each link state advertisement besides a default metric. This kind of an approach is also called as TOS-routing.

A special form of advertisements, called as Opaque LSA, is specified in [RFC2370].

The purpose of Opaque LSAs is to provide a generalized mechanism to extend the OSPF protocol in the future. There are three different types of Opaque LSAs in terms of the range of topological distribution, also referred to as the flooding scope. The flooding scopes are link-local scope (type 9), area-local scope (type 10) and flooding throughout the AS (type 11). An example where these kind of advertisements are in use is explained in Section 5.2.5.

5.2.2 Flooding link state information

In OSPF routing the routers exchange information about the link states and the metrics. This procedure has to be carried out carefully to avoid situations where the packets are routed back to the sender due to an incoherent link state database. Hello, exchange and flooding protocols are responsible for appropriate communication between different network elements. Hello protocol is used to check if the links are operational and to select a so-called designated router, which has special responsibilities in the link state update process in broadcast and non-broadcast networks. When two routers establish the two-way connectivity, the link state databases are synchronized by the exchange protocol. Finally, flooding protocol is activated when the link state changes occur.

The idea of the flooding protocol in its simplicity is the following: Router A detects a change in the status of some of its adjacent links and updates the link state database accordingly. After that router A sends link state advertisements to its neighbors. The advertisements include a time stamp to make a difference between old and new information. As router B receives the advertisement it checks the corresponding record in the link state database. If the record is missing or is older than the received one, router B adds the new advertisement to the database and broadcasts the message forward. If the received advertisement includes older information than the corresponding record in the database, router B transmits recent information back to the sender.

5.2.3 Computation of routes

Based on the information stored in the link state database, a graph of the network can be formed. The OSPF routers and the transit networks are considered as internal nodes and the stub networks, the summarized networks, and the external destinations act as edge nodes of the graph. Naturally, the arcs of the graph consist of links with varying link metrics, which are indicated by the TOS-field of the database. Based on the graph, each router constructs a tree consisting of the shortest paths to all destinations with itself as the root.

The calculation of the paths in OSPF is based on the ”shortest path first” al-

gorithm developed by E. W. Dijkstra. Let E be the set of evaluated nodes for which the shortest path is already calculated and set R consists of the rest of nodes. The paths are collected to set O. Each path has a cost corresponding to the links' metrics, such as the sum of the lengths of the links used by the path. Following the description presented in [Hui95] the algorithm runs as follows:

1. Insert source node s to E and the rest of the nodes to R. Insert all one-hop paths starting from s to O. Sort O in terms of increasing cost.
2. If O is empty, all nodes in R are unreachable and the algorithm terminates.
3. Let P be the first path in O (shortest path) and V be the last node in P. Remove P from O. If V is already in E go to step 2. Otherwise, P is the shortest path to V and V can be moved from R to E.
4. Update O by concatenating P and Vs adjacent nodes and inserting resulting paths to ordered list O. Go to step 2.

If the router enables TOS-routing, it has to compute the shortest paths using all TOS metrics, not only the default one. As the shortest path tree is formed at each router, the router can decide the correct next hop for the incoming packets and pass this information to the separate IP forwarding unit.

5.2.4 Equal Cost Multiple Path

Often one can find many paths with an equal shortest length connecting two routers in the network. In simple routing protocols, as in RIP, only one path is randomly selected to route traffic. However, it is often the case that utilizing all shortest paths would provide smoother traffic distribution in the network and some congestion situations would be avoided. In OSPF the link metrics are constrained to be integers with the range from 1 to 65535, which increases the probability of finding equal cost paths. The modifications to the shortest path algorithm are minor; a shortest path to node V is kept as an acceptable alternative path if distance from the source to the node preceding node V is shorter than the distance from the source to V.

When multiple shortest paths are used the question is how the incoming packets are distributed to those paths. This problem is outside the scope of OSPF specification [RFC2328]. It is just said that the traffic is split equally among the shortest paths. However, as in [For00] and [Sri03], we assume that equal splitting refers to the case where traffic is split equally to the allowed next hops, not equally to the parallel paths. In addition, the implementation of the exact traffic splitting is not a simple task.

Three mechanisms to split traffic are listed in [Vil99b]. Those mechanisms are:

1. Per packet round robin
2. Division of the destination prefixes among the available next hops
3. Division of traffic according to some hash function applied to the source and destination address pair.

The problem in the round robin is that, to avoid the performance collapse of TCP, the delays of different paths have to be close to each other, which is rarely true in the networks. Dividing packets based on the destination prefixes, for one, is a coarse and unpredictable technique to obtain equal split. According to [Vil99b], the most applicable mechanism to distribute traffic is hashing based on the source and destination addresses. In this approach the hash space is evenly split among the available paths and the hash function, such as CRC-16, is applied to the source and destination addresses.

5.2.5 TE extensions of OSPF

The routing by the OSPF shortest path calculation does not provide sufficient capabilities to serve different types of traffic properly and to avoid congestion situations in the network. Thus it is meaningful to make some modifications to OSPF routing that enable traffic engineering.

Recent modifications to OSPF routing to support traffic engineering are presented in RFC 3630 [RFC3630]. 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. Methods in [RFC3630] cover only intra-area distribution of traffic engineering information. In practise, the modifications are done to type 10 Opaque LSAs.

New LSAs, also called as traffic engineering LSAs, have a standard LSA header but the payload enables transmission of additional information. Payload of the LSA consists of one or more nested Type/Length/Value (TLV) fields. The LSAs are categorized by Type field, which can get two values, router address (1) or link (2). In the router address TLV, a stable IP address of the advertising router can be indicated, whereas in the link TLV different characteristics of the link, such as maximum bandwidth and unreserved bandwidth, can be announced to the routers of the network.

5.3 Static optimization of OSPF networks

Traffic engineering based on the OSPF weights started with the observation that the link weights can be changed by the network operator. For example, the weights can be changed by "ip ospf cost"-command in the Cisco operating system or by some network management system [For02b].

In OSPF the default value of the link weights is 1. Using unit link weights the hop-count is minimized and the shortest path routing minimizes the usage of the resources. Cisco has recommended to set the link weights inversely proportional to the link's capacity. By this choice the shortest path routing favors the link which probably has enough capacity to carry traffic.

The method to find a good link weight set depends on the way how traffic can be split to the parallel shortest paths. If only equal splitting is allowed, the weight setting problem is NP-hard. On the other hand, if unequal splitting is possible, the optimal weights can be found using the primal-dual method.

In this section we first present the network model of OSPF traffic engineering problems. After that we review the research related to the NP-hard weight setting problem and the problem of finding the optimal set of OSPF-weights when the unequal split is allowed.

5.3.1 Network model

Consider an IP network based on OSPF-routing (OSPF-network). As in Chapter 2, 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. Let $\mathcal{P}_k$ denote the set of all possible paths p from node s_k to node t_k. We use the notation $l \in p$ if link l belongs to path p. The traffic demand of IE-pair k is denoted by d_k.

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 from node s_k to node t_k with respect to link weights w_l,

$$\mathcal{P}_k^{\mathrm{SP}} = \{p \in \mathcal{P}_k \mid \sum_{l \in p} w_l = \min_{p' \in \mathcal{P}_k} \sum_{l' \in p'} w_{l'}\}.$$

In each node i, the incoming traffic with the same destination t is aggregated and then split to links (i, j) that belong to some shortest path of the ingress-egress pair (i, t). Such adjacent nodes j are called admissible next hops. Let ϕ_{ij}^t denote

the corresponding splitting ratios. Thus, ϕ_{ij}^t 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

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

For an illustration, see Figure 5.1. As, e.g., in [For00], it is usually assumed that these splitting ratios ϕ_{ij}^t are equal,

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

This choice is referred to as Equal Cost Multiple Path (ECMP) (see Section 5.2.4). However, there are methods that allow unequal splitting ratios, such as in [Vil99b] and [Sri03].

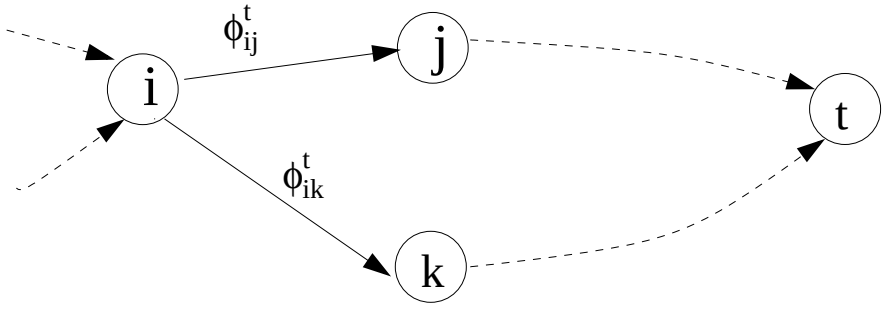

Figure 5.1: The OSPF network model

It may happen that the load for the bottleneck links becomes too high. Then, one naturally asks whether it is possible to choose the link weights w_l and the splitting ratios ϕ_{ij}^t in a clever way such that the optimization objectives are fulfilled. Indeed, this question can be viewed as a static load balancing problem in the context of OSPF networks.

5.3.2 Algorithms for finding a good weight set

Work of Fortz and Thorup

One of the first proposals to tune OSPF-weights to achieve an optimal routing was presented by B. Fortz and M. Thorup in [For00]. The objective of the paper is to study if the OSPF weights can be defined so that the optimal routing is achieved. In that work the optimal routing is defined as a solution of problem (2.1), where link cost C_l is a piece-wise linear, increasing and convex cost function. The assumption in the paper is that the traffic demand matrix can be estimated based on the measurements done in the backbone network. In addition, it is

assumed that traffic is split equally among the adjacent links that are on the shortest paths to the destination.

The first result of the paper is that the answer to the question stated above is negative: In general it is not possible to select link weights so that problem (2.1) is optimized. Even finding a best weight setting is a NP-hard problem. However, in paper [For00] a local heuristic search method is presented for finding the OSPF-weights so that the performance of a network is very close to that of optimal routing strategy (implemented using MPLS for example).

The local search heuristics for finding the optimal link weights in [For00] has two main steps: finding a candidate set of feasible solutions and selecting a next iterate from that candidate set. Let $W := \{1, 2, ..., w_{max}\}$ be the set of all possible link weights and $w = (w_l)_{l \in L}$ be the current solution vector for the weights. The next iterate w' is selected from a neighborhood of w, denoted by $\mathcal{N}(w)$. The neighborhood is constructed using two operations:

1. One single weight w_l is replaced by each possible weight $w' \in W \backslash \{w_l\}$ and other weights remain unchanged.

2. For a given source x and destination t, the weights of the links adjacent to node x are adjusted so that the lengths of all possible paths from x to t are equal. By this operation the number of shortest length paths is increased and, due to splitting to equal length paths, the traffic is distributed smoother in the network. An additional constraint in this operation is that, if the link load is over a random threshold, the weight remains unchanged.

Selecting the next iterate from the neighborhood is an essential task in the algorithm. If only solutions that improve the objective function are chosen, the algorithm stops in the local minimum for sure. Therefore, the algorithm of Fortz and Thorup allows non-improving moves by selecting randomly the new set of solutions from the candidates. Evaluation of the same link weight combinations many times is avoided by use of hashing tables.

The results of the paper show that traditional OSPF routing is competitive as compared with explicit routing protocols, such as MPLS. However, the paper does not point out clearly how the benefits are gained as the routing is compared with minimum hop-count routing. Are there usually many equal cost paths to the destinations or is the gain achieved by avoiding the congested paths to the destination by appropriate link weight setting? The large test-network obviously provides flexibility without traffic splitting.

In [For02a] the same authors point out that weight changes should be avoided as much as possible, because weight updates confuse the active routing and the performance of TCP goes down. Also a human network operator should be able

to control routing updates. The paper proposes that the optimization of the network is done by changing only a few OSPF weights at a time. The operator can limit the weigh changes if they do not improve the performance enough or they bring out some new problems. Daily fluctuations in traffic demand matrix are compensated for searching a weight setting that suits for many demand matrices. This is done by constructing a convex combination of the presentative demand matrices and finding a good weight setting for that.

The papers presented above assume that the traffic demand matrix is known. However, as discussed in Section 3.5, obtaining the point-to-point traffic matrix is not a straightforward task. Thus Roughan et al. continue the work of previous authors in [Rou03] by combining the traffic matrix estimation to traffic engineering. The paper estimates the traffic matrix by a generalized gravity model and then optimizes the OSPF weights by the algorithm of paper [For00]. The practical finding is that this combination yields to the results that are close to the optimum, when the maximum link utilization is minimized.

Genetic algorithms

As an outgrowth of the work of Fortz and Thorup a wide range of algorithms to find good or even best OSPF weight setting have been developed. One type of those algorithms are *genetic algorithms*, which are introduced in OSPF context by Ericsson et al. in [Eri02]. A basic principle of genetic algorithms is to transform a population of solutions, which have some value of objective function, into a new generation according to Darwin principle of survival. The idea of the iterative method of [Eri02] is following: Let $P(0)$ be the initial population, whose individuals are selected randomly, and $P(t)$ be the population at time t. At each iteration t, assign a fitness value (based on the objective function) to each individuals in $P(t)$, select the parents to mate and create children by crossover and mutation. Finally, select best individuals for the next iteration $t + 1$.

The algorithm reviewed above is further developed in [Bur02] resulting in a so-called *memetic algorithm* (memetic algorithm refers to population-based heuristic search in optimization problems). The difference to the genetic algorithm of Ericsson is minor; at each iteration the weights of k worst links in terms of the objective function are increased in order to move traffic to the less loaded links.

5.3.3 Methods for defining both optimal link weights and splitting ratios

Also papers [Wan01b] and [Sri03] investigate the weight setting problem. However, the assumption concerning the traffic splitting to multiple shortest paths

differs fundamentally from the algorithms in Section 5.3.2; instead of equal splitting, arbitrary splitting ratios are allowed.

The important result of [Wan01b] 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 found as a solution to the dual of the linear problem. This procedure is described in the following section in more detail.

As said above, the paper assumes that the traffic of each origin-destination pair can be split arbitrarily to the shortest paths. However, in the current IP routing with OSPF this is not possible. First, only equal splitting is standardized, and second, the splitting in each router is done based on the destination address only.

Paper [Sri03] tries to solve the disadvantages of paper [Wan01b]. The source-destination based splitting is easy to convert to destination based splitting. The splitting ratio of router i to destination s along link (i, j) is the sum of the traffics whose destination is s and which use link (i, j) divided by the sum of all traffic incoming to i with the same destination. The problem of unequal splitting ratios is solved by taking advantage of the existence of multiple prefixes to a certain destination. For a particular prefix, only a part of next hops are available. As the size of the routing table increases, this approach approximates well the arbitrary splitting ratios of the optimal routing.

5.3.4 Primal-Dual method for optimizing the link weights

In this section we first review the well-known duality problem in general and then consider the duality of the static load balancing problem. We discuss the role of duality in OSPF traffic engineering by reviewing the primal-dual method of Wang et al. in [Wan01b].

Formulation of duality for general optimization problem

Paper [Baz93] considers a non-linear optimization problem P, which is called the primal problem:

$$\begin{aligned} &\text{Minimize } f(\mathbf{x}) \\ &\text{subject to the constraints} \\ &g_i(\mathbf{x}) \leq 0 \qquad \text{for all } i = 1, ..., m, \\ &h_i(\mathbf{x}) = 0 \qquad \text{for all } i = 1, ..., l, \\ &\mathbf{x} \in X. \end{aligned} \tag{5.1}$$

The dual problem refers to a group of optimization problems closely related to the primal problem. In the dual problem minimization is changed to maximization and the side constraints are involved in the optimization function as "penalty". Lagrangian dual formulation is the best known dual problem, and is as follows:

$$\begin{aligned} &\text{Maximize } \theta(\mathbf{u}, \mathbf{v}) \\ &\text{subject to the constraints} \\ &\mathbf{u} \leq 0, \end{aligned} \tag{5.2}$$

where

$$\theta(\mathbf{u}, \mathbf{v}) = \inf\{f(\mathbf{x}) + \sum_i u_i g_i(\mathbf{x}) + \sum_i v_i h_i(\mathbf{x}) : \mathbf{x} \in X\}.$$

The formulation of the dual of the given primal problem is not unique since the constraints of the primal problem can be included both to inequalities $g(\mathbf{x}) \leq 0$ and equalities $h(\mathbf{x}) = 0$ and to constraint $\mathbf{x} \in X$.

Formulation of duality for the static load balancing problem

Next we present a dual problem for the static load balancing problem presented in Section 2.3.1. We consider only the problem where the path set is unconstrained. In problem (2.1) there are three constrains, upper and lower constraints for the link loads and the flow conservation constraint. The last one includes constraints for each IE-pair k and node i. Thus we introduce a penalty p_i^k for each IE-pair k and node i. The Lagrangian dual formulation is as follows:

$$\begin{aligned} &\text{Maximize } \theta(\mathbf{p}) \\ &\text{subject to no constraints on } p \end{aligned} \tag{5.3}$$

where

$$\theta(\mathbf{p}) = \inf\{C(x) + \sum_{k\in\mathcal{K}} \sum_{i\in\mathcal{N}} (A_i x^k - R_i^k) p_i^k : 0 \le x_l \le b_l\} \tag{5.4}$$

Dual problem to find the link weights

Wang et al. [Wan01b] proved that there is a set of positive link weights w_l so that the optimal paths in the load balancing problems (2.1) or (2.2) are shortest paths. The procedure for defining these link weights utilize the duality formulation of the previous subsection and is given below. The network model is the same as described in Subsection 5.3.1.

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 some load balancing problem presented in Section 2.3. Formulate then another LP-problem (primal) and its dual. In the primal LP-problem the induced traffic loads $\tilde{y}_l$ serve as new capacity constraints:

$$\begin{aligned}
&\text{Minimize } \sum_{k\in\mathcal{K}} \sum_{l\in\mathcal{L}} x_l^k \\
&\text{subject to the constraints} \\
&x_l^k \ge 0, && \text{for each } k \in \mathcal{K} \text{ and } l \in \mathcal{L}, \\
&\sum_{k\in\mathcal{K}} x_l^k \le \tilde{y}_l, && \text{for each } l \in \mathcal{L}\ , \\
&Ax^k = R^k, && \text{for each } k \in \mathcal{K},
\end{aligned} \tag{5.5}$$

Let U_i^k be the penalty associated to each IE-pair k and node i. The dual of the problem above is:

$$\begin{aligned}
&\text{Maximize } \sum_{k\in\mathcal{K}} d_k U_{t_k}^k - \sum_{l\in\mathcal{L}} \tilde{y}_l W_l \\
&\text{subject to the constraints} \\
&W_l \ge 0, && \text{for each } l \in \mathcal{L}\ , \\
&U_{s_k}^k = 0, && \text{for each } k \in \mathcal{K} \\
&U_j^k - U_i^k \le W_{(i,j)} + 1, && \text{for each } k \in \mathcal{K} \text{ and } (i,j) \in \mathcal{L}.
\end{aligned} \tag{5.6}$$

The *complementary slackness* property states that, for every used link of each IE-pair, the difference of the penalties between the nodes equals the link cost, that is,

$$U_j^k - U_i^k = W_{(i,j)} \text{ for each } x_l^k > 0.$$

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, the 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 [Sri03]:

$$\phi_{ij}^t = \frac{\sum_{k:t_k=t} x_{(i,j)}^k}{\sum_{j':(i,j')\in\mathcal{L}} \sum_{k:t_k=t} x_{(i,j')}^k} \tag{5.7}$$

5.3.5 Optimization of the splitting ratios

The network optimization by changing the link weights is often too time consuming or impractical. But even with fixed link weights we can still affect the traffic distribution by optimizing the traffic splitting ratios used in the routers.

We present a procedure for determining the splitting ratios that minimize the maximum link utilization with the given link weights. As before, let $\mathcal{P}_k^{\mathrm{SP}}$ denote the set of shortest paths for IE-pair k with respect to these link weights. Let ϕ_p denote the fraction of traffic demand d_k that uses path $p \in \mathcal{P}_k^{\mathrm{SP}}$. We start by solving these splitting ratios for each IE-pair k from the LP-problem, which is a variant of problem (2.9):

$$\begin{aligned}
&\text{Minimize } \alpha + r \sum_{k\in\mathcal{K}} \sum_{l\in\mathcal{L}} \sum_{p\in\mathcal{P}_k^{\mathrm{SP}}:l\in p} d_k\phi_p \\
&\text{subject to the constraints} \\
&\alpha \geq 0,\ \phi_p \geq 0, && \text{for each } p \in \bigcup_{k\in\mathcal{K}} \mathcal{P}_k^{\mathrm{SP}}, \\
&\sum_{k\in\mathcal{K}} \sum_{p\in\mathcal{P}_k^{\mathrm{SP}}:l\in p} d_k\phi_p \leq \alpha b_l, && \text{for each } l \in \mathcal{L}, \\
&\sum_{p\in\mathcal{P}_k} \phi_p = 1, && \text{for each } k \in \mathcal{K}.
\end{aligned} \tag{5.8}$$

Let ϕ_p be the optimal traffic share on path p. This induces the following link loads:

$$x_l^k = \sum_{p\in\mathcal{P}_k^{\mathrm{SP}}:l\in p} d_k\phi_p.$$

The destination based splitting ratios for each node i can then be calculated as in (5.7).

$$\phi_{ij}^t = \frac{\sum_{k:t_k=t} \sum_{p\in\mathcal{P}_k^{\mathrm{SP}}:(i,j)\in p} d_k\phi_p}{\sum_{j':(i,j')\in\mathcal{L}} \sum_{k:t_k=t} \sum_{p\in\mathcal{P}_k^{\mathrm{SP}}:(i,j')\in p} d_k\phi_p} \tag{5.9}$$

5.4 Adaptive approaches for load balancing

Many mechanisms are developed to improve the performance of the network by changing the link weights and thus the shortest paths adaptively. One of those proposals uses a bandwidth estimation method to compute new link weights to be distributed [Per02] and another attempts to minimize the packet loss rate by optimizing the OSPF weights by online simulation [Tah03].

However, as explained in [For02a] and in Section 5.3.2, it is not desirable to modify the link weights too frequently, since this can lead to undesired effects in the network performance and in addition, the optimization of link weights requires lots of computation. Thus the adaptive approaches for load balancing adaptively have to rely on some other mechanisms than setting the OSPF weights. One mechanism is to split traffic among the shortest paths unequally and thus achieve a desired traffic distribution in the network. We have presented the static optimization problem of adjusting the splitting ratios in Section 5.3.5.

In this section we review some adaptive mechanisms developed recently to OSPF network optimization.

5.4.1 OSPF-OMP

OSPF Optimized Multipath (OSPF-OMP) described in [Vil99b] is a mechanism that extends OSPF to support traffic-aware routing. An important note of [Vil99b] is that traffic can be split to the shortest paths unequally with a fine granularity by dividing hash space in different proportions. By this approach it is easier to affect the traffic distribution when compared with changing the OSPF weights.

In OSPF-OMP the decisions to change the division of hashing space are based on the flooded link load information. Link load information is carried in Opaque LSAs (see Section 5.2.5) and distributed by OSPF's standard flooding protocol. The information field of specific Opaque LSAs includes a measure of the link load as a fraction of link capacity, a measure of packet loss due to queue overflow, and the link capacity in kbits/s. These measures are computed by interface counters maintained for SNMP usage. LSAs are flooded only when a defined change between previous and the current "equivalent load" is detected. Computation of the equivalent load is based on estimated loss of TCP, that is, $equivLoad = load * K * \sqrt{loss}$, where K is chosen to 10.

Load distribution adjustments on multiple paths are done as follows: For every set of paths one link or part of paths, which have the highest equivalent load, is searched and that part is identified as a critically loaded segment. Load is adjusted by moving traffic from the paths that include critically loaded segments

to other paths according to "move increments", which are defined dynamically.

Superiority of OSPF-OMP depends on the ability to split traffic on the multiple shortest paths. Since it is difficult to set the link weights so that many of those paths can be found, OSPF-OMP relaxes the constraint of shortest path routing by allowing also paths for which next router is closer than the current one from the destination.

5.4.2 Other approaches

The purpose of Adaptive Multi-Path (AMP) algorithm proposed in [Goj03] is to reduce the memory consumption and signalling overhead of OSPF-OMP algorithm introduced in the previous section. As OSPF-OMP, AMP establishes multiple paths to each destination by allowing also the use of non-shortest paths and balances load by changing the splitting ratios at each router. However, in AMP a node has only local information of its adjacent links, which is the main difference to OSPF-OMP, where the global link load information is stored at each node. The algorithm performs as follows: If congestion on link (i, j) is detected, the end node i tries to shift traffic away to less loaded links. In addition, the node in the question sends backpressure messages (BMs) to its adjacent nodes $k \neq j$ to inform that they have caused congestion on link (i, j) by sending traffic to i. However, in paper [Goj03] it remains unclear how much the use of only local information affects the performance of AMP as compared with OSPF-OMP.

There has not been so much research on how to route elastic traffic. Usually traffic-aware routing tries to finds paths for stream traffic with bandwidth and delay guarantees. The main performance metric of an elastic flow is the average throughput, i.e. the ratio of average flow size to average transmission time. Also the blocking probability has an influence on service quality.

The idea of the routing algorithm in [Oue99] is to emphasize paths with widest bandwidth when load is light and shortest paths when the traffic load is heavy. The path with maximum utility is selected. The utility function is $U[P, B] = \ln r - B.h$, where B is a penalty constant, r is the fair share rate allocated to incoming flow and h presents the number of extra hops. Rate r can be calculated using formula $r = (m.MR)/(n + 1)$, where n is the current flow count metric, m is maximum flow count metric and MR is the minimum rate of the elastic flow. If $B = 0$ the algorithm is identical to the widest path algorithm and if $B \longrightarrow \infty$ the algorithm corresponds to the shortest path algorithm.

5.5 Proposal for distributed and adaptive load balancing in OSPF networks

A drawback of TE methods reviewed in Section 5.3 is that the traffic demand matrix is assumed to be known. In addition, the approaches do not take the traffic fluctuations into account. The studies related to adaptive approach in Section 5.4 do not provide a proper comparison of the approaches to min-hop routing and optimality, and the analysis of the stability issues is missing.

Therefore in this section we study how load can be balanced without the information about the traffic matrix. The adaptive algorithm to be presented in this section is developed from the corresponding algorithm presented in Section 4.4. We also study how a change of the link weights by the primal-dual method affects the robustness of the routing. The used network model is described in Section 5.3.1.

5.5.1 Dynamic load balancing problem

Similar to the MPLS networks, the static load balancing problem presented in Section 5.3 can be formulated and solved only if the traffic demands d_k are known. However, deriving the traffic matrix discussed in Chapter 3 is still an open question. If the traffic matrix is missing, another approach is needed. In this Section we first formulate the dynamic load balancing problem for OSPF-networks and then introduce the required changes to the adaptive and distributed algorithm of Section 4.4 to solve the dynamic problem.

Our assumptions are as follows. The traffic demands d_k are fixed but unknown. The link loads are periodically measured at times t_n. Let $\hat{y}_l(n)$ denote the measured link load of link l from the measurement period (t_{n-1}, t_n). The information on the measured loads is distributed to all nodes in the network. The time needed to distribute the information, typically 5 seconds, is negligible in comparison with the length of the measurement period.

In general, the objective of our dynamic load balancing problem is as follows. Based on the measured link loads, the link weights w_l and the destination based traffic splitting ratios ϕ_{ij}^t should be adjusted so that they converge, as soon as possible, to the (unknown) optimal values of the corresponding static load balancing problem discussed in Section 5.3.

However, as mentioned in Section 5.4, it is not desirable to modify the link weights too frequently. Therefore, we consider the dynamic problem at two time scales. At the shorter time scale, only the traffic splitting ratios are adjusted but the link weights are kept fixed. The objective in this case is to adjust the traffic

splitting ratios so that they converge to the (unknown) optimal values of the corresponding restricted optimization problem presented in Subsection 5.3.5. At the longer time scale, also the link weights should be adjusted so that the optimal load distribution is finally achieved. One way to do it is to estimate the traffic demands from the measurement data and then determine the link weights as a solution to the dual problem presented in Subsection 5.3.4.

In this thesis we focus on dynamic load balancing to minimize the maximum link utilization (problem (2.8)) at the shorter time scale. An adaptive and distributed algorithm to solve this problem is described in the following subsection. However, the algorithm can easily be changed to optimize other objectives also.

5.5.2 Adaptive and distributed algorithm for load balancing in OSPF networks

We assume that the link weights w_l are fixed. For each IE-pair k, let $\mathcal{P}_k^{\mathrm{SP}}$ denote the set of shortest paths from node s_k to node t_k with respect to these link weights w_l.

Let $\phi_{ij}^t(n)$ denote the traffic splitting ratios that are based on the measured link loads $\hat{y}(n) = (\hat{y}_l(n); l \in \mathcal{L})$. We note that, since the measured link loads are distributed to all nodes, the decisions concerning the traffic splitting ratios can be made in a distributed way. Thus, in our adaptive and distributed algorithm, each node i independently determines the traffic splitting ratios ϕ_{ij}^t for all destination nodes t and admissible next hops j.

The decisions in the algorithm are based on a cost function $D_p(y)$ defined for each path $p \in \mathcal{P}_{(i,t)}^{\mathrm{SP}}$ by

$$D_p(y) = \max_{l \in p} \frac{y_l}{b_l},$$

where $y = (y_l; l \in \mathcal{L})$. This is a natural choice as the objective is to minimize the maximum link utilization. The idea in the algorithm is simply to alleviate the congestion on the most costly path by reducing the corresponding traffic splitting ratio. This should, of course, be compensated for by increasing the splitting ratio related to some other path. A problem in the adaptive adjustment of the splitting ratios at a short time scale is the possible disordering of the packets. However, this can be solved by changing only a part of the splitting ratios at a time, for example.

To alleviate the oscillation problems founded already in ARPANET, we let the splitting ratios change only with minor steps. The step size is determined by the granularity parameter g. A finer granularity is achieved by increasing the value of g. The measurement period of a couple on minutes should be short enough to obtain reasonably fast convergence.

Algorithm At time t_n, after receiving the information $\hat{y}(n)$ concerning all the measured loads, node i adjusts the traffic splitting ratios for all destination nodes t as follows:

1. Calculate the cost $D_p(\hat{y}(n))$ for each path $p \in \mathcal{P}^{\mathrm{SP}}_{(i,t)}$.

2. Find the path $q \in \mathcal{P}^{\mathrm{SP}}_{(i,t)}$ with maximum cost, i.e. $D_q(\hat{y}(n)) = \max_{p \in \mathcal{P}^{\mathrm{SP}}_{(i,t)}} D_p(\hat{y}(n))$, and decrease the splitting ratio of the first link (i,j) of that path as follows:

$$\phi^t_{ij}(n) = \phi^t_{ij}(n-1) - \frac{1}{g}\phi^t_{ij}(n-1).$$

3. Choose another path $r \in \mathcal{P}^{\mathrm{SP}}_{(i,t)}$ randomly and increase the splitting ratio of its first link (i,k) as follows:

$$\phi^t_{ik}(n) = \phi^t_{ik}(n-1) + \frac{1}{g}\phi^t_{ij}(n-1).$$

4. For all other admissible next hops j', keep the old splitting ratio,

$$\phi^t_{ij'}(n) = \phi^t_{ij'}(n-1).$$

5.5.3 Evaluation method

Next we evaluate numerically the performance of the proposed adaptive load balancing algorithm in OSPF networks. First, we develop an evaluation method that follows the corresponding one in Chapter 4. Thereafter, in Section 5.5.4, the results of application of this evaluation method to two different test networks are presented.

The evaluation method is iterative and runs as follows. The test network includes the nodes n, links l, IE-pairs k, and paths p. The link weights w_l are first fixed. The traffic fluctuations are modelled by adding a random component to each deterministic traffic demand d_k as in Chapter 4. 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 $p \in \mathcal{P}^{\mathrm{SP}}_k$ 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

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

where the $\epsilon_k(n)$ are independent Gaussian random variables with mean 0 and variance $\delta^2 d_k^2$ describing the random fluctuations of traffic during measurement period n around the fixed demands d_k. As a result, we have a traffic model similar to [Cao00]. The coefficient of variation, δ, for this random variable is assumed to be the same for all IE-pairs k. After this, the new traffic splitting ratios $\phi_{ij}^t(n)$ are determined from the measured loads $\hat{y}_{(i,j)}(n)$ as presented in Subsection 5.5.2.

5.5.4 Numerical results

The test networks used in the evaluation process are the same as in the MPLS context in Section 4.4. The first network has 10 nodes, 52 links, and 72 IE-pairs and the second one has 20 nodes, 102 links, and 380 IE-pairs.

Three different traffic scenarios are used. In the first one, the random traffic fluctuations at the time scale of measurements are ignored by setting the fluctuation parameter δ to 0. In the second one, these random fluctuations are taken into account by setting the fluctuation parameter δ to 0.1. In the third scenario, we consider the traffic fluctuations at longer time scales by letting the traffic demands to change drastically three times during the evaluation period (after 500, 1000 and 1500 iterations, correspondingly).

The results of the adaptive algorithm are compared with

1. "ECMP": the standard policy where the traffic is split equally to the shortest paths with the unit link weights,
2. "Sub-optimal": the optimal value of the restricted optimization problem (5.8) with link weights fixed to 1, and
3. "Optimal": the optimal value of the static load balancing problem (2.8).

No traffic fluctuations

In this scenario $\delta = 0$. The shortest paths needed for the adaptive algorithm are calculated using link weights $w_l = 1$ for all links l. Figures 5.2 and 5.3 show the resulting maximum link utilization for the 10-node and 20-node networks as a function of the number of iterations for granularity parameters $g = 20$ and $g = 50$. We can see that the performance of the adaptive algorithm approaches

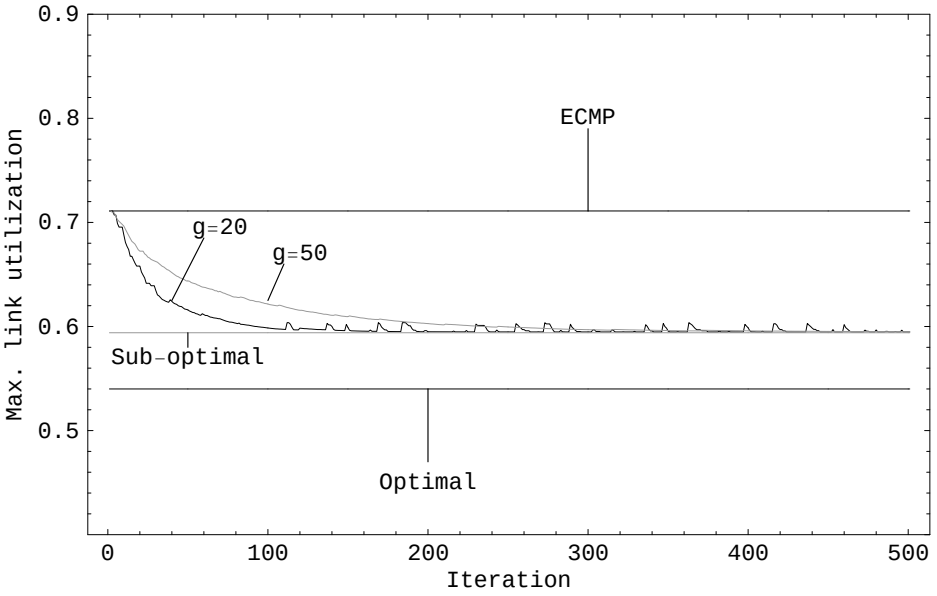

Figure 5.2: The maximum link utilization as a function of the number of iterations, when there are no traffic fluctuations, $\delta = 0$. 10-node network.

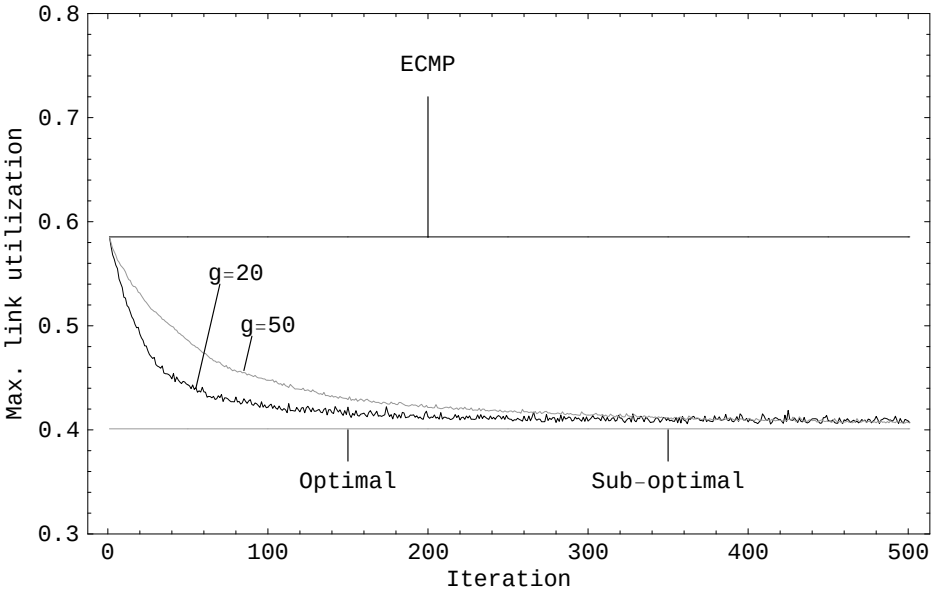

Figure 5.3: The maximum link utilization as a function of the number of iterations, when there are no traffic fluctuations, $\delta = 0$. 20-node network.

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. The convergence times are only two times as great as in the 20-node network (approx. 200 iterations) than in the 10-node network (approx. 100 iterations) in spite of the huge growth in the complexity of the network.

Random traffic fluctuations at a shorter time scale

In this scenario $\delta = 0.1$. The shortest paths needed for the adaptive algorithm are again calculated using link weights $w_l = 1$ for all links l. Figures 5.4 and 5.5 show the resulting maximum link utilization for the 10-node and 20-node networks as a function of the number of iterations for granularity parameters $g = 20$ and $g = 50$. We find that the random traffic fluctuations at a time scale of the measurement period induce oscillations to the maximum link utilization. However, even with granularity parameter $g = 20$, oscillations are tolerable and the algorithm converges close to the sub-optimal value.

Traffic fluctuations at a longer time scale

In this scenario the traffic demands change drastically three times. The shortest paths needed for the adaptive algorithm are first calculated using link weights $w_l = 1$ for all links l. Figures 5.6 and 5.7 show the resulting maximum link utilization for the 10-node and 20-node networks as a function of the number of iterations for granularity parameters $g = 20$ and $g = 50$. The adaptive algorithm reacts to the changes and provides a result close to the sub-optimal value in the 10-node network and close to the optimal value in the 20-node network. Note that in the 10-node network in the iteration rounds from 500 to 1000 the result of heuristics and the sub-optimal value are equal.

Optimization of both link weight and splitting ratios

In this subsection we assumed that the link weights and the splitting ratios of each router can be optimized at a longer time scale (hours to days). The shortest paths needed for the adaptive algorithm are thus first calculated using the optimal link weights corresponding to the original traffic demands d_k and determined from the dual problem (5.2) and the splitting ratios from the solution of the primal problem (5.1) and formula (5.7). After that the link weights remain unchanged. Figures 5.8 and 5.9 show the resulting maximum link utilization for the 10-node and 20-node networks as a function of the number of iterations for granularity parameters $g = 20$ and $g = 50$. Now the sub-optimal values in the 10-node

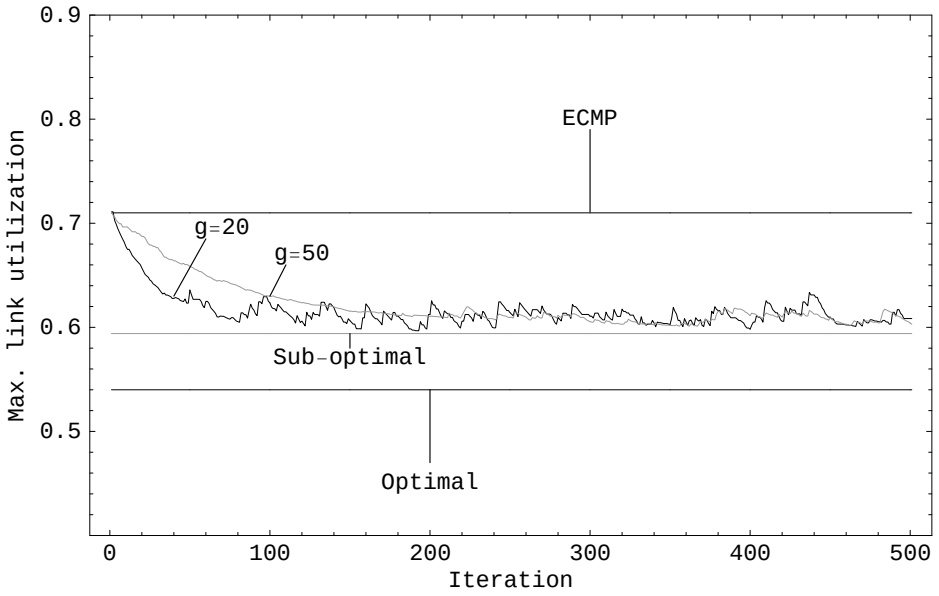

Figure 5.4: The maximum link utilization as a function of the number of iterations, when there are random traffic fluctuations at a shorter time scale, $\delta = 0.1$. 10-node network.

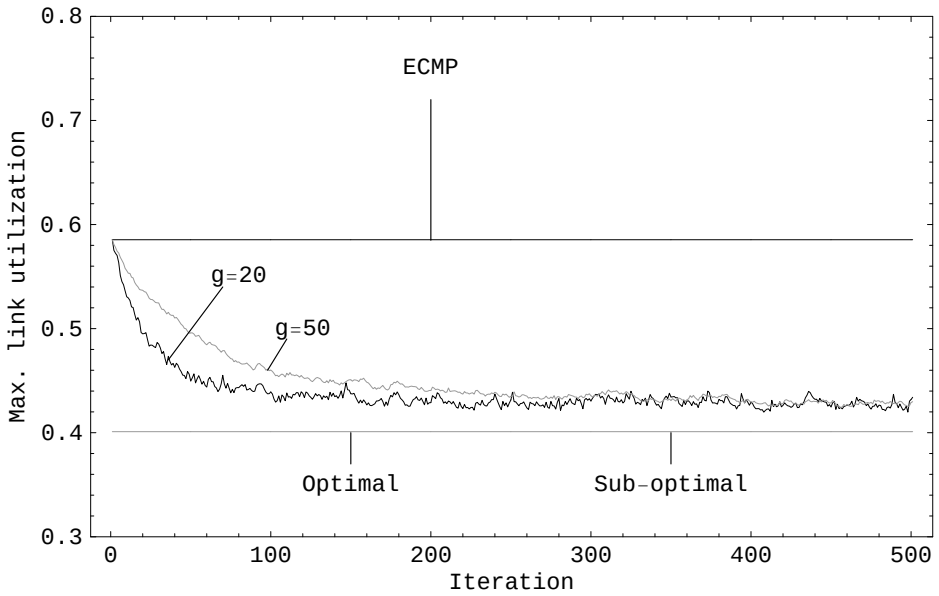

Figure 5.5: The maximum link utilization as a function of the number of iterations, when there are random traffic fluctuations at a shorter time scale, $\delta = 0.1$. 20-node network.

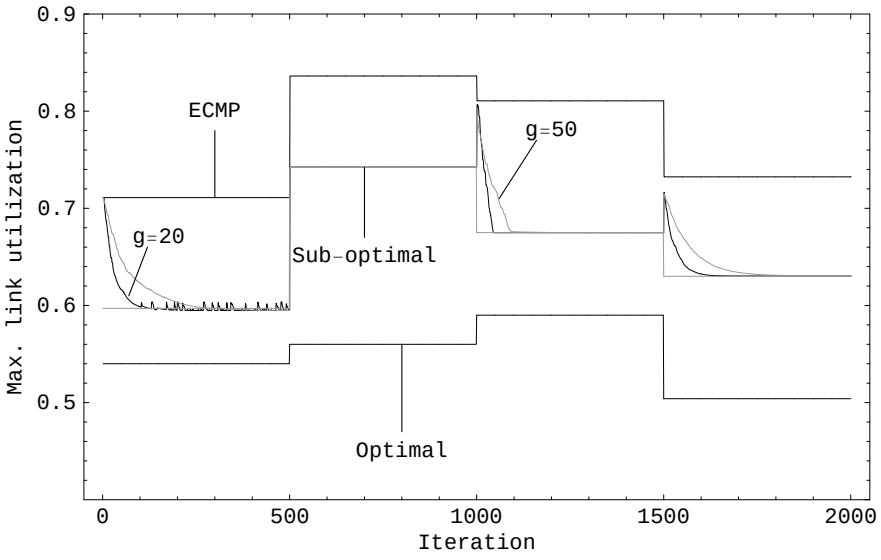

Figure 5.6: The maximum link utilization as a function of the number of iterations, when there are traffic fluctuations at a longer time scale. 10-node network.

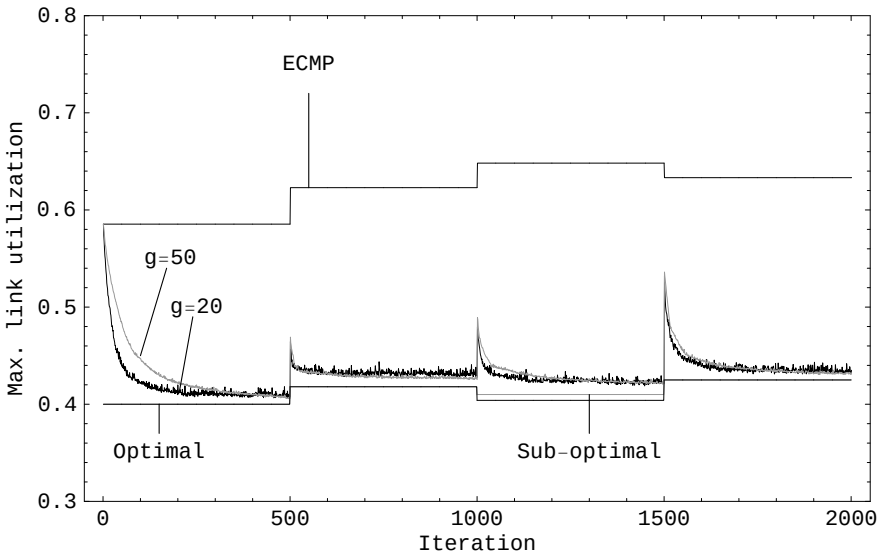

Figure 5.7: The maximum link utilization as a function of the number of iterations, when there are traffic fluctuations at a longer time scale. 20-node network.

network are close to the optimal values and thus heuristics can yield results close to the optimal solution. In the 20-node network the results are similar to the previous case in Figure 5.7, except for the first iterations, where the change of the link weight and the splitting ratios provide optimal results.

As a conclusion, the optimization of the link weights at the longer time scale improves the performance of the network. In the 20-node network also the unit weights provide a good result. An explanation is that, in the 10-node network, the number of shortest paths related to the unit link weights is 116 whereas it is 153 in the case of the optimal link weights. Thus the number of ϕ-parameters is greater in the latter case and also the results are better. In the 20-node network, the corresponding numbers of the shortest paths are 782 and 785. Also the results are quite similar.

We have also studied a variant of the algorithm in which traffic is moved from the congested direction to the shortest path which minimizes the path cost (compare with the variant MINCST in MPLS networks). However, the results were not promising; the algorithm converged many times to a sub-optimal value, which is far from the optimum.

5.6 Conclusion of TE in OSPF networks

In this chapter we studied how load can be balanced adaptively in OSPF-networks using a distributed approach. The main difference to the algorithm developed for MPLS networks is that decision parameters are now destination based, whereas they were source-destination based in MPLS networks. We also considered the procedure of optimizing the OSPF-weights by primal-dual method and how this can be combined with adaptive heuristics. The results show that the optimization of traffic splitting ratios improves the performance of the network when compared with equal splitting. When the set of shortest paths is small also the changing of OSPF-weights is worthwhile.

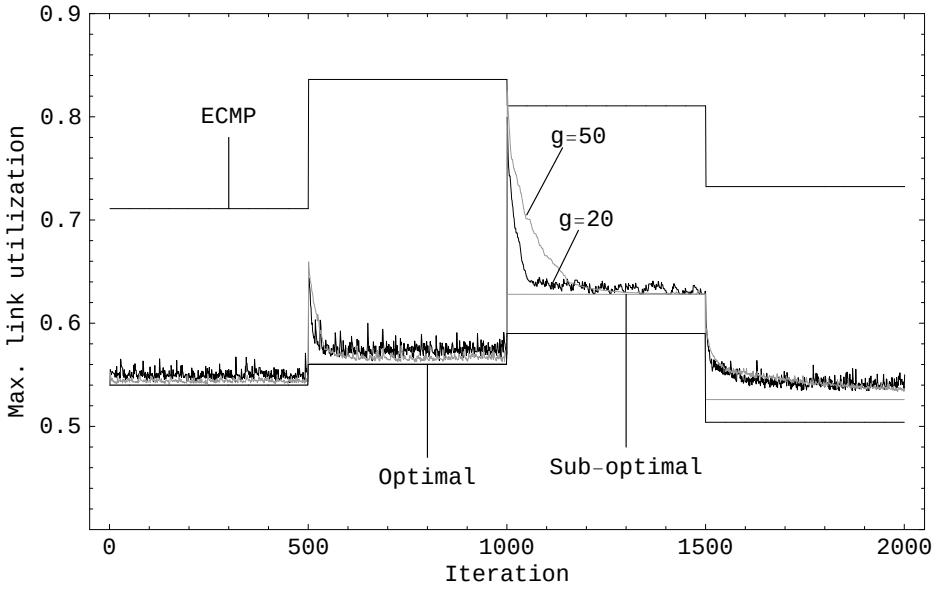

Figure 5.8: The maximum link utilization as a function of the number of iterations. 10-node network.

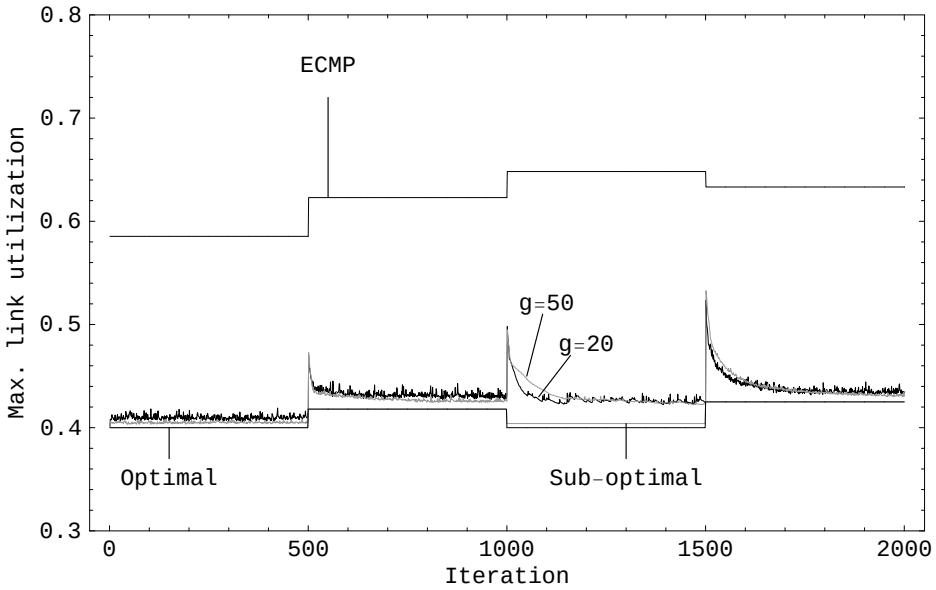

Figure 5.9: The maximum link utilization as a function of the number of iterations. 20-node network.

Chapter 6

Comparison of traffic engineering in different networks

This section compares the traffic engineering in MPLS and OSPF networks. First we concentrate on the traffic engineering capabilities in general and then we compare the effectiveness of our own adaptive algorithm in different platforms.

6.1 Pros and cons of different architectures

In this thesis we have studied traffic engineering 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. Those pros and cons are listed for both MPLS and OSPF networks.

The benefits of MPLS are:

- The support for the explicit paths is a part of MPLS technology. Those paths provide much flexibility to control the traffic traversing the network from the source to the destination.
- The use of explicit paths provide also a tool for measuring the traffic conditions network-wide. This is essential in performing traffic allocation optimization in the network.

The benefits of OSPF are:

- The shortest-path routing is spread to almost all networks and can be viewed as a standard technology.

- By a proper link weight and splitting ratio selection the optimal traffic allocation can be obtained.

Correspondingly, the disadvantages of the MPLS approach are:

- MPLS is not yet spread to be a de facto network standard.
- 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.

Finally, below are listed the disadvantages of the OSPF based traffic engineering:

- Load cannot be forwarded to the paths with different costs.
- To balance load without traffic splitting is hardly possible. Even, when equal split to the shortest paths is in used, the problem of finding the best OSPF-weights is NP-hard.
- In the destination-based traffic engineering the number of splitting ratio parameters is $N \cdot N$, where N is number of nodes. Thus the amount of overhead can be huge.
- If a link weight is changed, it is difficult to forecast the effects caused to the network performance.

As a conclusion, traffic engineering in MPLS based networks is easier. However, as the study of adaptive approach in Section 5.5 shows, the performance of the OSPF networks can be improved significantly by clever splitting ratio selection compared with the equal splitting in ECMP.

6.2 Convergence of the algorithms

In this section we compare the iteration times of the adaptive MPLS and OSPF algorithms proposed in Section 4.4 and 5.5, respectively. The objective is the same in both cases, minimization of the maximum link utilization in 10-node and 20-node networks with the granularity parameter fixed to $g = 20$.

From Table 6.1 we can see that the convergence times in MPLS networks are significantly shorter. In addition, the algorithm in OSPF networks converges to the sub-optimal value, because only the shortest paths are allowed. One reason for the difference is again the number of used paths. In the 10-node MPLS network

there are 286 LSPs, whereas in the OSPF network there are 116 shortest paths. In addition, the source-destination based decision parameters seem to work more effectively than the destination based parameters.

Table 6.1: The approximative number of iterations required to the convergence.

	RNDPTH (MPLS)	MINCST (MPLS)	OSPF
10-node network	40	20	100
20-node network	25	20	150

If 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.

Chapter 7

Conclusions

This section gives a short summary of the thesis and discusses further research directions.

7.1 Summary

Best-effort service of the Internet is not adequate anymore as new type of applications emerge to be available worldwide. Measurements of the Internet traffic indicate that often some links are congested while others remain lightly loaded. Thus implementation of some traffic engineering mechanism is reasonable.

This thesis has studied traffic engineering in the Internet from many perspectives. We have reviewed the static load balancing problem with different optimization objectives and some algorithms for solving the problem. We have also discussed adaptive traffic engineering and general scepticism towards adaptive routing in the Internet. The requirements for adaptive traffic engineering mechanisms to overcome these problems are identified. Also different measurement alternatives to be used in traffic engineering are covered.

We have introduced both MPLS architecture and OSPF routing to provide a technical platform for traffic engineering. In addition, both static and adaptive load balancing algorithms developed recently for these networks are reviewed.

The main contribution of this thesis is the development of simple adaptive and distributed algorithms for load balancing in MPLS and OSPF networks. The assumption is that traffic matrix is not available and thus the balancing is based on the measured link loads only. The idea is to make incremental changes in the traffic distribution and thus avoid undesirable oscillations. In MPLS networks,

the traffic distribution of the network is controlled in the edge routers by changing the ingress-egress based splitting ratios. Similarly, in OSPF networks each router makes traffic engineering decisions by adjusting the destination based splitting ratios.

We have evaluated the algorithms in both types of networks numerically by developing a simple evaluation method. The results show that the performance of the network can be effectively improved even when the traffic demands fluctuate at the time scale of the measurement period. One difference in the results of MPLS and OSPF networks is that in an MPLS network the algorithm converges to the optimum much faster than in an OSPF network.

7.2 Further research directions

In this thesis we have evaluated the adaptive algorithms using only simulated link loads. In the future, the purpose is to use real measured data obtained from operating backbone networks. The appropriate length for the measurement period has to be studied in more detail. The period should be short enough so that the algorithm reacts rapidly to changes in the traffic matrix. It would also be useful to utilize the higher moments of the measurements in the load balancing algorithm. In the current version the balancing is based on the average link loads only.

The mechanisms for traffic engineering should be extended to cover also multiple routing areas. If it is possible to control which egress router is selected (compare with the third traffic model in Subsection 2.2.2) the static optimization problem should be reformulated. In addition, the possibilities to balance the load by configuring the BGP routing parameters should be studied.

Bibliography

[And03] E. J. Anderson, T. E. Anderson, On the Stability of Adaptive Routing in the Presence of Congestion Control, in Proceedings of IEEE INFOCOM, 2003.

[App03] D. Applegate and M. Thorup, Load optimal MPLS routing with N+M labels, in Proceedings of IEEE INFOCOM, 2003.

[Arm00] Armitage, MPLS: The Magic Behind the Myths, IEEE Communications Magazine, January 2000.

[Ash00] G. R. Ash, Traffic Engineering & QoS methods for IP-, ATM-, & TDM based multiservice networks, IETF Internet draft, <draft-ietf-tewg-qos-routing-00.txt>, 2000.

[Ban01] A. Banerjee, J. Drake, J. P. Lang, and B. Turner, Generalized Multiprotocol Label Switching: An Overview of Routing and Management Enhancements, IEEE Communications Magazine, Vol. 39, Issue 1, January 2001.

[Baz93] M. Bazaraa, H. Sherali and C. Shetty, Nonlinear Optimization: Theory and Algorithms, John Wiley & Sons., 2. edition, 1993.

[Bea97] N.G. Bean, F. P. Kelly and P. G Taylor, Braess' Paradox in a Loss Network, in the Journal of Applied Probability, 1997.

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

[Ber92] D. Bertsekas and R. Gallager, Data Networks, Prentice Hall, Englewood Cliffs, N.J., 2nd edition, 1992.

[Bur02] 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, pp. 187-188, 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.

[Carp] T. Carpenter, K.R. Krishnan and D. Shallcross, Enhancements to Traffic Engineering for Multi Protocol Label Switching, Telcordia Technologies.

[Cao00] J. Cao, D. Davis, S. Vander Wiel and B.Yu, Time-varying Network Tomography: Router Link Data, Journal of the American Statistical Association, vol. 95, pp. 1063–1075, 2000.

[Cha00] S. Challinor, An introduction to IP networks, Carrier-scale IP networks, BT exact communications technology series 1, Vol 18, No 3, July 2000.

[Chu02] C.-N. Chuah and C. Diot, A Tier-1 ISP perspective: Desing principles & observations of routing behavior, in Proceedings of IPAM Workshop on Large-Scale Communication Networks, 2002.

[Din00] E. Dinan, D. Awduche, B. Jabbari, Optimal Traffic Partitioning in MPLS Networks, in Proceedins of Networking 2000.

[Elw01] A. Elwalid, C. Jin, S. Low and I. Widjaja, MATE: MPLS Adaptive Traffic Engineering, in Proceedings of IEEE INFOCOM, 2001.

[Eri02] 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, Vol. 6, no.3, pp.299-333, 2002.

[Fel01] A. Feldmann, A. Greenberg, C. Lund, N .Reongold, J. Rexford and F. True, Deriving Traffic Demands for Operational IP NEtowrks: Methodology and Experience. IEEE/ACM Transactions on Networking Vol.9, Nro. 3 (June 2001) pp. 265-279.

[For00] B. Fortz, M. Thorup, Internet Traffic Engineering by Optimizing OSPF Weights, in Proceedings of IEEE INFOCOM, March 2000.

[For02a] B. Fortz and M. Thorup, Optimizing OSPF/IS-IS Weights in a Changing World, IEEE Journal on Selected Areas in Communications, Vol. 20, No. 4, 1 (May 2002).

[For02b] B. Fortz, J. Rexford and M. Thorup, Traffic Engineering With Traditional IP Routing Protocols, IEEE Communication Magazine, 1 (Oct 2002).

[Fra03] 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/December 2003.

[Gal77] R. Gallager, A Minimum Delay Routing Algorithm Using Distributed Computation, IEEE Transactions on Communication, Vol. COM-25, Nr 1, pages 73-85, January 1977.

[Gha99] A. Ghanwani, B. Jamoussi, D. Fedyk, P. Ashwood-Smith, L. Li and N. Feldman, Traffic Engineering Standards in IP Networks using MPLS, IEEE Communications Magazine, December 1999.

[Goj03] I. Gojmerac, T. Ziegler, F. Ricciato and P. Reichl, Adaptive Multipath Routing for Dynamic Traffic Engineering, in Proceedings of IEEE Globecom, Nov. 2003.

[Gro02] M. Grossglauser, J. Rexford, Passive Traffic Measurement for IP Operations, a chapter in "The Internet as a Large-Scale Complex System", Oxford University Press, 2004..

[Güv04] T. Güven, C. Kommareddy, R J. La, M.A. Shayman, B. Bhattacharjee, Measurement Based Optimal Multi-path Routing, in Proceedings of IEEE INFOCOM, 2004.

[Hui95] C. Huitema, Routing in the Internet, Prentice Hall, 1995.

[Iov03] P. Iovanna, M. Settembre, R. Sabella, A Traffic Engineering System for multi-layer networks based on the GMPLS paradigm, IEEE Network, March 2003.

[Kod00] M. Kodialam and T.V. Lakshman, Dynamic Routing of Bandwidth Guarantees Tunnels with Restoration, in Proceedings of IEEE INFOCOM, 2000.

[Lai01] Wai Sum Lai, R. W. Tibbs, and S. Van den Berghe, Requirements for Internet Traffic Engineering Measurement, Internet draft <draft-ietf-tewg-measure-06.txt>, July 2003.

[Med02] A. Medina, N. Taft, K. Salamatian, B. Bhattacharyya and C. Diot, Traffic Matrix Estimation: Existing Techniques and New Directions, in Proceedings of SIGCOMM'02 (August 2002).

[Ott01] 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-26027, 2001.

[Oue99] S. Oueslati-Boulahia and E. Oubagha, An approach to Routing Elastic Flows, in Proceedings of ITC 16. (1999).

[Per02] T. B. Pereira and L. L. Ling, Network Performance Analysis of and Adaptive OSPF Routing Strategy-Effective Bandwidth Estimation, in International Telecommunication Symposium ITS 2002, Brazil.

[RFC1157] J. Case, M. Fedor, M. Schoffstall, J. Davin, Simple Network Management Protocol (SNMP), IETF RFC 1157, May 1990.

[RFC1633] R. Braden, D. Clark and S. Shenker, Integrated Services in the Internet Architecture: an Overview, IETF RFC 1633, June 1994.

[RFC2328] J. Moy, OSPF version 2., IETF RFC 2328, April 1998.

[RFC2370] R. Coltum, The OSPF Opaque LSA Option, IETF RFC 2370, July 1998.

[RFC2386] E. Crawley, R. Nair, B. Rajagopalan and H. Sandick, A Framework for QoS-based Routing in the Internet, IETF RFC 2386, August 1998.

[RFC2475] S. Blake et al., An Architecture for Differentiated Services, IETF RFC 2475, December 1998.

[RFC2676] G. Apostopoulos, D. Williams, S. Kamat, R. Guérin, A. Orda and T. Przygienda, QoS Routing Mechanisms and OSPF Extensions, IETF RFC 2676, August 1999.

[RFC2702] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell and J. McManus, Requirements for Traffic Engineering over MPLS, IETF RFC 2702, September 1999.

[RFC3031] E. Rosen, A. Viswanathan and R. Callon, Multiprotocol Label Switching Architecture, IETF RFC3031, January 2001.

[RFC3036] L. Andersson, P. Doolan, N. Feldman, A. Fredette and B. Thomas, LDP Specification, IETF RFC 3036, January 2001.

[RFC3209] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan and G. Swallow, RSVP-TE: Extensions to RSVP for LSP Tunnels, December 2001.

[RFC3212] R. Dantu, L. Wu, P. Doolan, T. Worster, N. Feldman, A. Fredette, M. Girish, E. Gray, J. Heinänen, T. Kilty and A. Malis, Constraint-Based LSP Setup using LDP, IETF RFC 3212, January 2002.

[RFC3270] F. Le Faucheur, L. Wu, B. Davie, S. Davari, P. Vääränen, R. Krishnan, P. Cheval and J. Heinänen, Multi-Protocol Label Switching (MPLS) Support of Differentiated Services, IETF RFC 3270, May 2002.

[RFC3272] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja and X. Xiao, Overwiew and Principles of Internet Traffic Engineering, IETF RFC 3272, May 2002.

[RFC3630] D. Katz, K. Kompella and D. Yeung, Traffic Engineering (TE) Extensions to OSPF Version 2, IETF RFC 3630, September 2003.

[Rou03] M. Roughan, M. Thorup and Y. Zhang, Traffic Engineering with Estimated Traffic Matrices, in Proceedings of IMC'03, 2003.

[Sav99] S. Savage, A. Collins, and E. Hoffman, The End-to-End Effects of Internet Path Selection, in Proceedings of ACM SIGCOMM (September 1999).

[Seo01] Y. Seok, Y. Lee and Y. Choi, Dynamic Constrained Multipath Routing for MPLS Networks, Computer Communications and Networks, 2001.

[Son03] J. Song, S. Kim, M. Lee, H. Lee and T. Suda, Adaptive Load Distribution over Multipath in MPLS Networks, in Proceedings of ICC'03, Anchorage, Alaska, May, 2003.

[Sri00] 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, Salvador, Brazil, 2002.

[Sri03] A. Sridharan, R. Guérin and C. Diot, Achieving Near-Optimal Tarffic Engineering Solutions for Current OSPF-IS-IS Networks, in Proceedings of IEEE INFOCOM, May 2003.

[Sus02] R. Susitaival, J. Virtamo and S. Aalto, Load balancing by MPLS in differentiated services networks, in Proceedings of International Workshop Art-Qos 2003, pp. 252-264, 2003.

[Sus04a] R. Susitaival, S. Aalto and J. Virtamo, Adaptive load balancing using MPLS, accepted for publication in Proceedings of MMB&PGTS, 2004.

[Sus04b] R. Susitaival and S. Aalto, Adaptive load balancing with OSPF, accepted for publication in Proceedings of HET-NETs'04, 2004.

[Tah03] H. T. Kaur, T. Ye, S. Kalyanaraman, K. S. Vastola, Minimizing Packet Loss by Optimizing OSPF Weights Using On-line Simulation, in Proceedings of the 11th International IEEE/ACM Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems. MASCOTS-2003, 2003.

[Tsi86] J. Tsitsiklis and D. Bertsekas, Distributed Asynchronous Optimal Routing in Data Networks, IEEE Transactions on Automatic Control, Vol. AC-31, No. 4, April 1986.

[Vil99a] C. Villamizar, MPLS Optimized Multipath (MPLS-OMP), Internet draft <draft-villamizar-mpls-omp-01>, February 1999.

[Vil99b] C. Villamizar, OSPF Optimized Multipath (OSPF-OMP), Internet draft<draft-villamizar-ietf-ospf-omp-02>, February 1999.

[Vis98] A. Viswanathan, N. Feldman, Z. Wang and R. Callon, Evolution of Multi-Protocol Label Swithing, IEEE Communication Magazine, pages 165-173, May 1998.

[Wan01a] Z. Wang, Internet QoS, Architecture and Mechanisms for Quality of Service, Morgan-Kaufman Publishers, USA, 2001.

[Wan01b] Y. Wang, Z. wang and L. Zhang, Internet Traffic Engineering without Full Mesh Overlaying, in Proceedings of IEEE INFOCOM, 2001.

[War52] J. Wardrop, Some Theoretical Aspects of Road Traffic Research, in Proceedings of Inst. Civil Engineers, Part 2 , 1952.

[Zee99] M. van der Zee, Quality of Service Routing, Open report, Ericsson, July 1999.