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

Riikka Susitaival

Load balancing by MPLS in differentiated services networks

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Engineering

Espoo, Finland, September 4, 2002

Supervisor: Professor Jorma Virtamo, HUT

Instructors: Ph.D. Pirkko Kuusela and Ph.D. Samuli Aalto

HELSINKI UNIVERSITY OF TECHNOLOGY Abstract of the Master's Thesis

Author:	Riikka Susitaival
Title of the thesis:	Load balancing by MPLS in differentiated services networks
Date:	September 4, 2002
Number of pages:	68+5

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

Supervisor:	Professor Jorma Virtamo
Instructors:	Ph.D. Pirkko Kuusela and Ph.D. Samuli Aalto

In IP routing, forwarding decisions are made independently in each router. Multi Protocol Label Switching (MPLS) as a new technique assigns a short label to each packet at the ingress node of MPLS network and packets are forwarded according to these labels. The most significant application of MPLS is the Traffic Engineering, which is used to optimize the performance of networks.

The capability of MPLS of explicit routing as well as of splitting of the traffic on several paths allows load balancing . The goal of this thesis is to study a flow allocation that minimizes the mean delay of the network. The thesis concentrates on the minimum-delay algorithm presented by Gallager and its two approximations. The first approximation defines the paths using the LP-optimization and after that allocates traffic using the NLP-optimization. The second approximation divides traffic into parts and routes them consecutively using Dijkstra's algorithm.

Using the load balancing algorithms as a starting point, the main goal of this thesis is to develop optimization algorithms that differentiate classes in terms of mean delay. In the thesis, the differentiation is achieved by the use of both routing and WFQ-scheduling. Both optimal and approximative algorithms are developed for the joint optimization of the WFQ-weights and routing.

The load balancing algorithms and the algorithms to differentiate classes are implemented and tested. As a result it is found that the use of the approximations simplifies the optimization problem but still provides results that are near to optimal. The algorithms that try to differentiate traffic classes by both routing and WFQ-scheduling provide the best performance.

Keywords: MPLS, load balancing routing, differentiated services, WFQ

TEKNILLINEN KORKEAKOULU Diplomityön tiivistelmä

Tekijä: Riikka Susitaival

Työn nimi: Kuormantasaus MPLS:n avulla eriytetyn palvelun verkoissa

Päivämäärä: 4. syyskuuta 2002
Sivumäärä: 68+5

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

Työn valvoja: Professori Jorma Virtamo
Työn ohjaajat: Ph.D. Pirkko Kuusela ja fil. tri Samuli Aalto

IP-verkossa reitityspäätökset tehdään itsenäisesti jokaisessa reitittimessä. Multi Protocol Label Switching (MPLS) on leimakytkentäprotokolla, joka liittää jokaiseen pakettiin leiman MPLS-verkon reunalla. Leiman avulla määritellään reititys MPLS-verkossa. MPLS:n merkittävin sovellus on Traffic Engineering, jota käytetään verkon suorituskyvyn optimointiin.

MPLS:n eksplisiittisellä reitityksellä sekä jakamalla liikennettä useille poluille voidaan tasata kuormaa. Työn tarkoituksena on tutkia liikenteenjakoa, joka minimoi verkon keskiviiveen. Työssä keskitytään Gallagerin esittämään, viiveen minimoivaan algoritmiin sekä kahteen sen approksimaatioon. Ensimmäinen approksimaatio määrittelee ensin polut lineaarisella optimoinnilla ja tämän jälkeen reitittää liikenteen epälineaarisella optimoinnilla. Toinen approksimaatio jakaa liikenteen osiin ja reitittää osat perätysten Dijkstran algoritmin avulla.

Työn tärkein päämäärä on kehittää optimointialgoritmeja, jotka eriyttävät luokkia viiveen mukaan. Kuormaa tasaavia algoritmeja käytetään lähtökohtana. Eriyttämistä saavutetaan sekä reitityksellä että WFQ-skeduloinnilla. Työssä kehitetään sekä optimaalisia että approksimoivia algoritmeja WFQ-painojen ja reitityksen yhtäaikaista optimointia varten.

Kuormaa tasaavat sekä palvelua eriyttävät algoritmit toteutetaan ja testataan. Havaitaan, että approksimaatiot pienentävät laskenta-aikaa merkittävästi kuitenkaan huonontamatta reitityksen suorituskykyä. Palvelua eriyttävistä algoritmeista parhaiten toimivat ne, jotka hyödyntävät sekä reititystä että WFQ-skedulointia.

Avainsanat: MPLS, kuormantasaus, palvelun eriyttäminen, WFQ

Preface

This thesis was carried out at the Networking Laboratory at Helsinki University of Technology as a part of the COST 279 project.

I would like to thank my supervisor, professor Jorma Virtamo, for his valuable guidance and ideas. I would also like to thank my instructor Pirkko Kuusela for her guidance in endless optimization problems and instructor Samuli Aalto for his persistent work to get the thesis completed. I would also thank my colleagues at the Networking Laboratory for friendly atmosphere. Especially, my office mates Laura Nieminen and Johanna Antila deserve thanks for supportive discussions.

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

Espoo, 4.9.2002,

Riikka Susitaival

Contents

Acronyms

- **AF** Assured Forwarding
- **AS** Autonomous System
- **ATM** Asynchronous Transfer Mode
- **BA** Behavior Aggregate
- **BGP** Border Gateway Protocol
- **BTT** Bidirectional Traffic Trunk
- **CBS** Committed Burst Size
- **CDR** Committed Data Rate
- **CR-LDP** Constraint Routing Label Distribution Protocol
- **DE** Default
- **DS** Differentiated Services
- **DS-TE** Diff-Serv aware MPLS Traffic Engineering
- **EF** Expedited Forwarding
- **EGP** Exterior Gateway Protocol
- **E-LSP** EXP-Inferred-PSC LSP
- **FEC** Forwarding Equivalence Class
- **FF** Fixed Filter
- **FFQ** Fluid Fair Queueing
- **FR** Frame Relay
- **GPS** Generalized Processor Sharing
- **IETF** Internet Engineering Task Force
- **IGP** Interior Gateway Protocol

- **ILM** Incoming Label Map
- **IP** Internet Protocol
- **IS** Integrated Services
- **IS-IS** Intermediate System-Intermediate System
- **ISP** Internet Service Provider
- **LDP** Label Distribution Protocol
- **L-LSP** Label-only-inferred-PSC LSP
- **LP** Linear Programming
- **LSP** Label Switching Path
- **LSR** Label Switching Router
- **NHLFE** Next Hop Label Forwarding Entry
- **MAM** Maximum Allocation Multiplier
- **MATE** MPLS Adaptive Traffic Engineering
- **MF** Multi-Field
- **MPLS** Multi Protocol Label Switching
- **NLP** Non-Linear Programming
- **OA** Ordered Aggregate
- **OSPF** Open Shortest Path First
- **PASTE** Provider Architecture for Differentiated Services and Traffic Engineering
- **PBS** Peak Burst Size
- **PDR** Peak Data Rate
- **PHB** Per-hop Behavior
- **PSC** PHB Scheduling Class
- **QoS** Quality of Service
- **RFC** Request For Comments
- **RIP** Routing Information Protocol
- **RSVP** Resource Reservation Protocol

- **SE** Shared Explicit
- **SLA** Service Level Agreement
- **SLS** Service Level Specification
- **TCA** Traffic Conditioning Agreement
- **TCP** Transmission Control Protocol
- **TCS** Traffic Conditioning Specification
- **TE** Traffic Engineering
- **TOS** Type Of Service
- **UDP** User Datagram Protocol
- **VC** Virtual Circuit
- **WDM** Wavelength-Division Multiplexing
- **WFQ** Weighted Fair Queueing

Chapter 1

Introduction

1.1 Background

With Internet Protocol (IP), datagrams are forwarded on packet-by-packet basis. In the conventional IP routing, forwarding decisions are made independently in each router, based on the packet's header and precalculated routing tables. On the other hand, Asynchronous Transfer Mode (ATM) and Frame Relay (FR) are widely used connection oriented technologies, which use virtual circuits that are set up by a signalling protocol beforehand. There are several technologies to run IP over ATM, but these techniques have remarkable scaling problems [Vis98, Arm00].

MPLS (Multi Protocol Label Switching) is a flexible technology that enables new services in IP networks and makes routing more effective. It combines two different approaches, datagram and virtual circuit, as a compact technology. MPLS is based on short fixed length labels, which are assigned to each packet at the ingress node of the MPLS cloud [RFC3031]. These labels are used to make forwarding decisions at each node. The assignments of a particular packet to a particular FEC (Forwarding Equivalence Class) are made only once. This simplifies and improves forwarding.

One of the most significant applications of MPLS is Traffic Engineering. Traffic Engineering (TE) concerns performance optimization of operational networks. It handles technological applications and scientific principles to the measurement, modelling, characterization and control of the Internet traffic and applications to achieve specific performance objectives [RFC2702].

Traffic Engineering using MPLS provides mechanisms to route traffic with fixed starting point and destination 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 ben-

efit of traffic splitting is the ability to balance the load. Load balancing reduces congestion and therefore improves the performance of the network.

There are several load balancing algorithms designed for various networks, such as ATM networks. These methods could be adapted to MPLS networks also. One example of a flow allocation algorithm balancing the load is as follows: Paths are first determined using LP-calculation by minimizing the maximum link load and, after that, traffic is allocated to these paths using NLP-calculation by minimizing the mean delay.

The IP routing paradigm offers services only on the best-effort basis. However, the demand to provide different services to different customers is evident. Quality of Service (QoS) is under a widespread discussion today. Two notable architectures, Integrated Services and Differentiated Services, have been introduced in the literature. These architectures provide principles and goals to develop current IP networking.

MPLS and its Traffic Engineering capabilities could provide technical support to the implementation of QoS. The differentiation of services can be obtained by an alternative flow allocation that has the same principles as the load balancing methods. In order to make differentiation more effective, scheduling mechanisms, like WFQ-scheduling, can be utilized in the same context.

1.2 Objectives

The first objective of the thesis is to study MPLS as a technique. We wish to find out how it works and what capabilities it provides. Also our purpose is to study Traffic Engineering, especially from the MPLS point of view.

MPLS provides capabilities to allocate traffic flexibly. Congestion can be avoided using effective flow allocation. The computation of an optimal flow allocation that minimizes the mean delay of the network is quite hard. Our objective is to search and compare flow allocation algorithms that provide optimal or near optimal solutions. The computation time is an important performance metric.

However, our most important objective is to develop flow allocation methods that differentiate traffic classes in terms of mean delay at the network level. Technical capabilities of the MPLS architecture, flow allocation algorithms and the principles of QoS techniques, like WFQ-scheduling, are meant to be utilized when the methods to differentiate classes are developed.

1.3 Structure of the thesis

The thesis is organized as follows: The second chapter starts with a brief introduction of current IP routing protocols. The architecture of MPLS is explained and the most important application of MPLS, Traffic Engineering, is introduced.

In the third chapter we study how paths can be computed and traffic allocated to the paths in an optimal manner. We introduce both optimal and near optimal algorithms.

The fourth chapter focuses on the quality of service. We introduce two proposed architectures, Integrated Services and Differentiated Services. Also scheduling, in particular WFQ-scheduling, is included in the fourth chapter. Finally, some QoS routing mechanisms and architectures to provide Differentiated Services using MPLS are briefly introduced.

In the fifth chapter we develop routing methods that try to differentiate the quality of service in terms of mean delay. The methods can be divided into two classes. The first class routes different traffic classes differently. Thus, the traffic classes achieve differentiated services. In the second class we make use of WFQ-scheduling when trying to differentiate traffic classes.

The results of both the optimal flow allocation methods and the flow allocation methods to achieve differentiated services are introduced in the sixth chapter. We study how the load of network and the level of differentiation affect the mean delay. The conclusions of the thesis are made in the seventh chapter, which also includes a discussion of further work.

Chapter 2

MPLS and Traffic Engineering

2.1 Introduction

Packets are forwarded independently in each router in current IP networks. The forwarding algorithm is based on the longest match, which means that the more specific prefix is preferred over the less specific prefix. However, the algorithm is quite slow. Also the route selection based on the shortest path restricts the capabilities to manage the resources of the network.

MPLS as a new technology tries to overcome the problems of traditional IP routing. With MPLS, a fixed length label is assigned to each packet and forwarding in the MPLS network is done according to this label. The full IP header analysis is done only once at the ingress node of the MPLS network, which simplifies and accelerates the routing process. MPLS supports two types of route selection, hop-by-hop routing and explicit routing.

The services in traditional IP network are not predictable. The strength of MPLS is in provisioning of Traffic Engineering capabilities. The use of explicit routes, for example, gives the ability to manage the network resources and support new services.

In the beginning of this chapter we introduce shortly some traditional IP routing protocols. After that we focus on the architecture of MPLS. The key elements of the architecture are explained in section 2.3.1. We introduce three Label Distribution Protocols proposed by IETF in section 2.3.3. The concept of Traffic Engineering is introduced in section 2.4.1 and the network model of Traffic Engineering in section 2.4.2. Section 2.4.3 describes the idea of the traffic trunk. We introduce the purpose of traffic trunk attributes and resource attributes in section 2.4.4. Finally, in section 2.4.5 we make some notes of constraint-based routing paradigm.

2.2 Conventional IP routing

In IP routing, the routers make independent decisions about how to forward packets. When a packet arrives, the router forwards it according to the 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 the 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). They maintain the view of the topology of a local network and calculate the best paths to the all addresses of the network.

RIP as a distance-vector routing protocol is suitable for small networks. With RIP, the cost information of paths to all destinations is periodically changed between neighboring routers. However, there are problems with RIP due to the use of single metric and limited hop distance.

In a large network application, OSPF as a link-state routing protocol is recommended. 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 its function is to exchange information between Autonomous Systems (ASs) by disseminating accessibility information.

2.3 MPLS architecture

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 an 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]

2.3.1 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 it presents a particular FEC to which packet is designed to forward. 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 [Wan01].

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. 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 2.1). LSP Ingress pushes a label on some depth m of packet's label stack. The routers between the ingress and egress nodes make forwarding decisions based on a label on the 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.

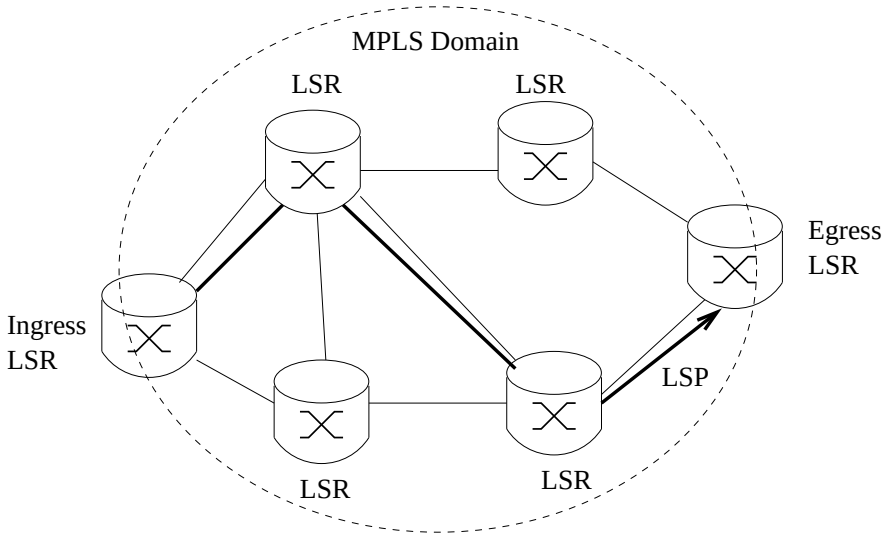

Figure 2.1: MPLS forwarding model

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.

Incoming Label Map

Incoming Label Map (ILM) (also known as 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 to 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 2.3.2) 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.

2.3.2 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.

2.3.3 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 done. 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. Label bindings by a downstream LSR can be done in two ways. In the first model the upstream LSR requests the neighbor LSR to bind a particular label for a particular FEC. In the second model downstream LSR can make bindings straight without demand. LDP is also used to exchange information of 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].

LDP

LDP is the first label distribution protocol proposed by IETF MPLS Working Group. LDP is meant to support hop-by-hop routing model. LDP includes four categories of messages. The first category includes messages to inform and maintain the appearance of LSR in the network. The messages of the second category are used to construct and maintain sessions between LSR peers. The third category includes messages to inform of the label mappings for FECs and the fourth category of messages is used to provide advisory and error information.

LSPs may be set up independently or sequentially from the egress node to the ingress node. In the first approach, when an LSR recognizes a particular FEC, it independently binds a label to that FEC and advertises the mapping to the neighbors. In the latter approach, an LSR can bind a label only to a FEC for which it already has the label mapping for the next hop FEC or for which it is an egress node. These types can be used at the same time in the same network.

Two types of label allocations are used with LDP: downstream allocation and downstream-on-demand allocation. A downstream node makes the label assignments in the first approach. In the latter approach an upstream LSR requests a downstream LSR to make label assignment.

CR-LDP

As Traffic Engineering has appeared to be the most significant application of MPLS, the requirement for a label distribution protocol that supports constraint-based routing has become important. Constraint-based routing label distribution protocol (CR-LDP) is proposed to extend LDP to enable constrained paths. Explicit routing can be viewed as a subset of constraint-based routing, where the

explicit route represents the constraint. The explicit routing gives an ability to use some other route than the route chosen by IP routing.

The difference between MPLS routing described earlier and constraint-based routing is that constraint-based routes are defined at the edge of the network. This enables many characteristics to be defined to the LSPs. The extensions to LDP proposed in CR-LDP are explicit routing, traffic characteristics, route pinning, CR-LSP preemption, failure handling, LSP ID and resource classes.

An explicit route is a set a LSRs and is identified by the LSP ID. There are two types of explicit routes, strict and loose. In the strict mode, each hop of a CR-LSP is associated with a unique IP address. The loose mode enables more choices in the route set up process, the route can contain abstract nodes (i.e. set of nodes) for example. The actual node to be used is determined locally from a set of nodes. LSPs are set up hop-by-hop starting from the ingress node. Every LSR sends a Label Request message forward to the next LSR until the egress node is reached. Then a Label Mapping message is sent backward from egress node.

The traffic characteristics of a particular path are peak data rate (PDR), peak burst size (PBS), committed data rate (CDR), committed burst size (CBS), service granularity and weight. The peak and committed data rate specify the bandwidth characteristics and the service granularity describes how the data rates are calculated. The weight describes the interrelationship between the excess bandwidth and the committed data rate.

Sometimes it is undesirable to change the path used by an LSP in the case of loose routes. Route pinning gives the ability to fix a particular node of the set of nodes even if there exists a better hop at some LSR.

Path preemption is used when the resources for a particular LSP cannot be found. An existing path may be rerouted in order to release resources to the new path. There are two priorities associated with an LSP, setup priority and holding priority. The path preemption is possible if the setup priority of the new path is higher than holding priority of the existing path.

Resources in the network can be classified. In the initiation phase of CR-LSP it is essential to specify which resource classes a particular path is able to use.

In the CR-LSP model, the states exist until they are explicitly removed. This approach is referred to as a hard-state system versus a soft-state system, where states are deleted after a particular time-out period has elapsed. The use of the hard-state system reduces overhead, because the LSPs need not to be refreshed. However, the use of time-out period helps to avoid error situations.

For further information, [RFC3213] discusses the applicability of CR-LDP and [RFC3214] introduces the approach to modify the parameters (e.g. bandwidth reservation) of existing CR-LSP using CR-LDP so that the modifications do not

interrupt the service providing.

RSVP-TE

A third label distribution protocol is based on Resource Reservation Protocol (RSVP). The original RSVP is extended to support Traffic Engineering and the model is referred to as RSVP-TE. RSVP-TE supports various new features like instantiation of explicitly routed LSPs with or without resource reservation, smooth rerouting of LSPs, preemption and loop detection.

LSPs can carry "traffic trunks". The concepts LSP and traffic trunk are distinct but only slightly. The use of traffic trunks allows certain attributes of the traffic to be parameterized. An LSP can carry many traffic trunks or conversely a traffic trunk can be routed to several LSPs in order to share load.

The concept of an LSP tunnel refers to the path set up by RSVP-TE to make difference to the path created by RSVP. An LSP tunnel is used to tunnel below normal IP routing. The use of LSP tunnels enables many different policies to optimize network performance, like establishing multiple LSPs between ingress and egress pairs.

Two reservation styles of RSVP are applied to the RSVP-TE. Fixed filter (FF) reservation style reserves resources separately to each sender. In the RSVP-TE context FF reservation style means that each ingress-egress pair has its own point-to-point LSP. In the shared explicit (SE) style the receiver can determine which senders are included in the reservation. In the RSVP-TE context this means that separate LSPs are created by assigning different labels to different senders [Wan01].

RSVP-TE is a soft-state system. After a time-out period, states are automatically deleted. So the protocol should refresh LSPs in order to keep them alive. This refreshing procedure can create a significant amount of overhead data and therefore there can be scalability problems in the RSVP-TE network.

2.4 MPLS Traffic Engineering

Traffic Engineering refers to performance optimization of operational networks at both the traffic and the resource levels. An overview of Traffic Engineering in IP networks is presented in [RFC3272].

MPLS can be used to provide Traffic Engineering. 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].

2.4.1 Internet Traffic Engineering

Internet Traffic Engineering consists of performance evaluation and optimization of operational IP networks. The major objective of Internet Traffic Engineering is the improvement of performance, both at the traffic level and at the resource level. Traffic oriented performance concentrates on the QoS of traffic streams. The minimization of packet loss and delay, the maximization of throughput and the 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 the bandwidth is the most scarce resource, so the major function is to manage bandwidth allocation.

Both in traffic and resource oriented objectives the minimization of congestion is crucial. The congestion cases 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 minimized using techniques provided by Traffic Engineering. Congestion can be avoided using load balancing policies. The objective can then be minimizing the maximum congestion. However, Traffic Engineering should be carried out in such a way that congestion can be managed cost-effectively.

The optimization of performance 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 (IGP) are Open Shortest Path First (OSPF) and Intermediate System-Intermediate System (IS-IS) [Gha99]. 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.

The support for equal-cost multipath [RFC2328] improves performance in the 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 the case where two different traffic streams are routed along the same link. In addition, the equal-cost multipath approach is not scalable.

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 can be developed to provide functionality of Traffic Engineering in the large networks also. The costs are lower, because MPLS provides capabilities 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 2.2.

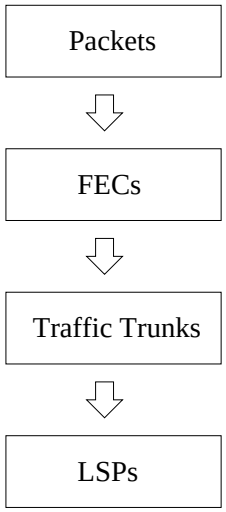

Figure 2.2: The hierarchy of MPLS streams

2.4.2 Induced MPLS graph

The concept of an induced MPLS graph is essential in Traffic Engineering over MPLS. An induced MPLS graph can be compared to a virtual topology of the overlay model. With the overlay model, a service provider establishes a virtual network that consists of all connections between edges. 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 is, according to [RFC2702], 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 link, 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.

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.

2.4.3 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 an 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 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 destroy operation. At the same time the resources like the label space and bandwidth are released.

2.4.4 Traffic trunk attributes and resource attributes

The behaviors 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 underlying protocol responses to the situations where traffic trunks become non-compliant. The attributes indicate, if 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.

2.4.5 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.

Chapter 3

Load balancing algorithms

3.1 Introduction

IP routing routes packets based on shortest path algorithms. It does not take into account the congestion of links when determining paths. However, using links that are congested can result in very long delays compared to an optimal flow allocation.

The possibility to use predetermined paths in routing allows one to allocate traffic effectively. The technologies where this approach is possible are ATM virtual circuits and MPLS, for example. Load balancing methods make an attempt to balance load in the network and therefore achieve better performance in terms of delay. The basic optimization problem minimizes the mean delay in the network. The use of the link delays of M/M/1-queues leads to a non-linear optimization problem (NLP). Many exact algorithms are introduced to this optimization, the most famous one being Gallager's algorithm from year 1977 [Gal77].

The flow allocation that minimizes the mean delay of network can be approximated in many ways. One coarse but simple approximation is to minimize the maximum link delay. This optimization function leads to a linear optimization program (LP), which is easier to solve than NLP-optimization.

In this chapter we study three flow allocation algorithms exactly. The first algorithm, introduced in chapter 3.2, is minimum-delay routing by Gallager [Gal77]. The algorithm results in an optimal flow allocation. The second algorithm, introduced in section 3.3, makes an attempt to reduce computation time of Gallager's routing method by solving paths by minimizing the maximum link load (LP-optimization) and after that by allocating traffic to paths using the mean delay as optimization function (NLP-optimization) [Ott01]. In section 3.4 we study the third algorithm that divides traffic into parts and allocates them using Dijkstra's algorithm [Srid00].

We study also some other methods briefly in section 3.5. We study paper [And01], which compare static routing algorithms with traffic-aware routing algorithms. Paper [Vut00] presents a model where the traffic that does not tolerate out-of-order delivery is routed first and the rest of the traffic is routed after that. Finally, we make some notes of MPLS Adaptive Traffic Engineering (MATE)-algorithm.

3.2 Minimum-delay routing

R. Gallager defines in [Gal77] an algorithm that minimizes the delay in a quasi-static network using distributed or centralized computation. The algorithm establishes the routing tables in the individual nodes based on the periodic information exchange between adjacent nodes. With successive updates of the routing tables, the average delay per message converges to the minimum average delay over all routing assignments. Traffic to each destination is guaranteed to be loop free at each iteration of the algorithm.

3.2.1 Formulation of minimum-delay routing

The formulation of the problem according to Gallager is as follows: Let L be the set of links, $L = \{(i,k) : \text{there exists a link from node } i \text{ to node } k\}$. Let $r_i(j)$ be the expected traffic (bit/s) entering the network at node i and destined for node j, $t_i(j)$ be the total expected traffic at node i and destined for node j, $\phi_{ik}(j)$ be the fraction of the node flow $t_i(j)$ that is routed over link (i,k) and $b_{(ik)}$ be the capacity of link (i,j). Since the node flow $t_i(j)$ at node i is the sum of input traffic and the traffic routed to i from other nodes,

$$t_i(j) = r_i(j) + \sum_l t_l(j)\phi_{li}(j), \quad \forall i,j. \tag{3.1}$$

Now let

$$f_{ik} = \sum_j t_i(j)\phi_{ik}(j) \tag{3.2}$$

be the expected traffic on link (i,k).

Theorem 1 (Gallager): Let a network have input set r and routing variable set ϕ. The set of equations (3.1) has a unique solution for t. Each component $t_i(j)$ is nonnegative and continuously differentiable as a function of r and ϕ.

Let D_{ik} be the expected number of messages/s transmitted on link (ik) times an expected delay/message. So it can be seen that the total expected delay per

message times the total expected number of message/s is given by

$$D_T = \sum_{i,k} D_{ik}(f_{ik}). \tag{3.3}$$

The assumption is that D_{ik} is an increasing and convex function of f_{ik}. Since the total message arrival rate doesn't depend on routing, the paper [Gal77] minimizes expected delay/message on the network by minimizing D_T over all choices of routing variables (i.e. set ϕ).

We present the optimization problem that minimize the mean delay of the network. The formulation of the problem using the notation presented above is as follows:

Minimize

$$E[D] = \frac{1}{\Lambda} \sum_{(ik)} \frac{f_{ik}}{b_{ik} - f_{ik}} = \frac{1}{\Lambda} \sum_{(ik)} \frac{\sum_j t_i(j)\phi_{ik}(j)}{b_{ik} - \sum_j t_i(j)\phi_{ik}(j)},$$

subject to the constraints

$$\begin{aligned} f_{ik} &= \sum_j t_i(j)\phi_{ik}(j) < b_{ik}, && \text{for each link } (ik), \\ t_i(j) &= r_i(j) + \sum_l t_l(j)\phi_{li}(j), && \text{for each } i, j, \end{aligned} \tag{3.4}$$

where Λ is the total offered traffic of the network.

3.2.2 Conditions for minimum delay

Firstly the partial derivatives of the total delay D_T with respect to the inputs r and the routing variables ϕ are taken in paper [Gal77]:

$$\frac{\partial D_T}{\partial r_i(j)} = \sum_k \phi_{ik}(j) \Big[D'_{ik}(f_{ik}) + \frac{\partial D_T}{\partial r_k(j)} \Big], \tag{3.5}$$

$$\frac{\partial D_T}{\partial \phi_{ik}(j)} = t_i(j) \Big[D'_{ik}(f_{ik}) + \frac{\partial D_T}{\partial r_k(j)} \Big]. \tag{3.6}$$

A necessary condition for a minimum of D_T with respect to ϕ are, for all $i \neq j$, $(i,k) \in L$,

$$\frac{\partial D_T}{\partial \phi_{ik}(j)} \begin{cases} := \lambda_{ij}, & \phi_{ik}(j) > 0, \\ \geqslant \lambda_{ij}, & \phi_{ik}(j) = 0. \end{cases} \tag{3.7}$$

This states that for given i, j, all links (i, k) for which $\phi_{ik}(j) > 0$ must have the same marginal delay $\frac{\partial D_T}{\partial \phi_{ik}(j)}$, and all links (i, k) for which $\phi_{ik}(j) = 0$ this marginal delay should be less than or equal to $\frac{\partial D_T}{\partial \phi_{ik}(j)}$. But (3.7) is not sufficient to minimize D_T.

Theorem 2 (Gallager): For each (i, k) assume that $D_{ik}(f_{ik})$ is convex and continuously differentiable for $0 \leq f_{ik} < C_{ik}$, where capacity $C_{ik} > 0$. Let ψ be the set of ϕ for which the link flows satisfy $f_{ik} < C_{ik}$ for all $(i, k) \in L$. Then (3.7) is necessary for ϕ to minimize D_T over ψ and

$$D'_{ik}(f_{ik}) + \frac{\partial D_T}{\partial r_k(j)} \geqslant \frac{\partial D_T}{\partial r_i(j)} \tag{3.8}$$

for all $i \neq j, (i, k) \in L$ is a sufficient condition.

Condition (3.8) states that in the optimal routing the marginal delay to node j through node k must be less than or equal to the marginal delay through some other node.

3.2.3 The algorithm

The idea of the algorithm of Gallager is that each node i must incrementally decrease those routing variables $\phi_{ik}(j)$ for which marginal delay $D'_{ik}(f_{ik}) + \frac{\partial D_T}{\partial r_k(j)}$ is large and increase those for which it is small. The algorithm is divided into two parts, one part is a protocol, which calculates the marginal delays, and the other part is an algorithm, which modifies routing variables. A routing variable set ϕ is loop-free if for each destination i, there is no m $(m \neq i)$ such that i is both upstream and downstream from m.

The first part of algorithm, the protocol to calculate the marginal delays is as follows: For each destination node j, each node i waits until it has received the value $\frac{\partial D_T}{\partial r_k(j)}$ from each of its downstream neighbors $k \neq j$. Then node i calculates $\frac{\partial D_T}{\partial r_i(j)}$ from (3.5) (using the convention that $\frac{\partial D_T}{\partial r_j(j)} = 0$) and sends this information to all of its neighbors except destination node j. If this procedure is loop free, it is also deadlock free.

The second part of the algorithm that modifies the current routing variable set ϕ to a set ϕ^1 is as follows: Let $B_i(j)$ be the set of blocked nodes k (i.e. $\phi_{ik}(j) = 0$) and

$$\phi^1_{ik}(j) = 0, \quad \Delta_{ik}(j) = 0, \qquad \forall k \in B_i(j). \tag{3.9}$$

The difference between the marginal delay to node j using link (ik) and the marginal delay to node j using an optimal link is calculated:

$$a_{ik}(j) = D'_{ik}(f_{ik}) + \frac{\partial D_T}{\partial r_k(j)} - \min_{m \notin B_i(j)} \Big[D'_{im}(f_{im}) + \frac{\partial D_T}{\partial r_m(j)} \Big], \qquad \forall k \notin B_i(j). \tag{3.10}$$

After that the amount of reduction in routing variables, denoted by $\Delta_{ik}(j)$, is calculated using the equation

$$\Delta_{ik}(j) = \min \left[\phi_{ik}(j), \eta a_{ik}(j)/t_i(j) \right], \tag{3.11}$$

where η is a scale parameter. Denote that $\Delta_{ik}(j)$ is inversely proportional to $t_i(j)$. Let $k_{min}(i,j)$ be a value of m that achieves the minimization in (3.10). Finally, the fractions of traffic are moved from the non-optimal links to the optimal link using the following equation:

$$\phi^1_{ik}(j) = \begin{cases} \phi_{ik}(j) - \Delta_{ik}(j), & k \neq k_{min}(i,j), \\ \phi_{ik}(j) + \sum_{k \neq k_{min}(i,j)} \Delta_{ik}(j), & k = k_{min}(i,j). \end{cases} \tag{3.12}$$

The changes in routing parameters and thus the speed convergence of the algorithm depends on the scale parameter η.

3.3 Two-level procedure for flow allocation problem

The goal of paper [Ott01] is to find an optimal multi-commodity flow allocation. The approach is to use both linear programming (LP) and non-linear programming (NLP) to solve the problem. LP-computation solves which paths to use and NLP-computation defines an optimal flow allocation to these paths. The time consuming LP-problem is calculated infrequently off-line, but the NLP-problem is calculated continuously. The NLP algorithm used is a version of the algorithm explained in Section 3.2.

3.3.1 Linear programming formulation and solution

The formulation of the problem presented in [Ott01] is as follows: A network consists of N nodes. A pair of nodes (m,n) can be connected by a directed link

(m,n) with bandwidth equal to $b_{(m,n)}$. The number of links is denoted by L and the topology is denoted by T, which is a set of node-pairs. The traffic demands are given by the matrix $[d_{(i,j)}]$, where i is the ingress and j is the egress node. Demands and capacities are assumed to be constant (or average traffic rates). Let $x_{(i,j),(m,n)}$ be the allocated traffic of ingress-egress pair (i,j) on link (m,n). Then the total traffic on the link (m,n) is

$$X_{(m,n)} = \sum_{(i,j)} x_{(i,j),(m,n)}. \tag{3.13}$$

A slack $S_{(m,n)}$ is defined in [Ott01] by

$$S_{(m,n)} = b_{(m,n)} - X_{(m,n)}. \tag{3.14}$$

If traffic fluctuations should be taken into account, optimization could be considered as max-min criterion, when the objective is to maximize

$$\min_{(m,n)} \Big(\frac{S_{(m,n)}}{C_{(m,n)}} \Big), \tag{3.15}$$

where $C_{(m,n)}$ is a positive coefficient. Let Z denote the maximal value of (3.15). If $C_{(m,n)} = \sqrt{b_{(m,n)}}$, the solution minimizes the maximal probability of link flow exceeding its bandwidth.

The routing that allocates traffic load $X_{(m,n)}$ on link (m,n) is *non-dominated* if there does not exist another routing with link-loads $X'_{(m,n)}$ with property

$$X'_{(m,n)} \le X_{(m,n)} \text{ for all } (m,n) \text{ and } X'_{(m,n)} < X_{(m,n)} \text{ for at least one } (m,n). \tag{3.16}$$

Max-min criterion (3.15) has the disadvantage that the optimal solution is not non-dominated. It does not take into account a high hop-count or large delays. That is the reason why paper [Ott01] presents a combined parametric criterion. The pair-based flow formulation of the linear optimization presented in [Ott01] is as follows:

$$\begin{aligned}
&\text{Minimize } \Big[-\epsilon Z + \sum_{(m,n)} w_{(m,n)} \sum_{(i,j)} x_{(i,j),(m,n)} \Big] \\
&\text{subject to the constraints} \\
&x_{(i,j),(m,n)} \ge 0; \quad Z \ge 0, \\
&\sum_{(i,j)} x_{(i,j),(m,n)} + C_{(m,n)} Z \le b_{(m,n)}, && \text{for each } (m,n) \text{ with } b_{(m,n)} > 0, \\
&\delta_{i,n} d_{(i,j)} + \sum_{k} x_{(i,j),(k,n)} = \delta_{n,j} d_{(i,j)} + \sum_{k} x_{(i,j),(n,k)}, && \text{at each node } n,
\end{aligned} \tag{3.17}$$

where $\delta_{(i,j)} = 1$ when $i = j$ and $\delta_{(i,j)} = 0$ otherwise.

The last equation in (3.17) defines that, at every node n, incoming traffic of each ingress-egress pair must be equal to outgoing traffic. The equation can also be

formulated as in [Cas94]. Let $A \in \mathbb{R}^{N \times L}$ be the matrix, for which $A(i,j) = -1$ if link j directs to node i, $A(i,j) = 1$ if link j leaves from node i and $A(i,j) = 0$ otherwise. $R_{(i,j)} \in \mathbb{R}^N$ is a row vector for each ingress-egress pair (i,j) such that $R_{(i,j),k} = d_{(i,j)}$, if k is the ingress node, $R_{(i,j),k} = -d_{(i,j)}$, if k is the egress node and $R_{(i,j),k} = 0$ otherwise. $x_{(i,j)} \in \mathbb{R}^L$ is a flow vector for each ingress-egress pair (i,j). Now condition

$$Ax_{(i,j)}^T = R_{(i,j)}^T, \quad \text{for each } (i,j) \tag{3.18}$$

assures the paths for each ingress-egress pair. This notation is used in further calculations.

3.3.2 Defining paths from the LP-solution

When the LP-problem (3.17) is solved and variables $x_{(i,j),(m,n)}$ found, we have to find paths for each ingress-egress pair. However, the solution is not unique. Consider a network example presented in Figure 3.1, where node 1 is the ingress node, node 7 is the egress node and the traffic allocation on each link is 100. Now there are many possibilities to select paths. Two examples are presented in Table 3.1.

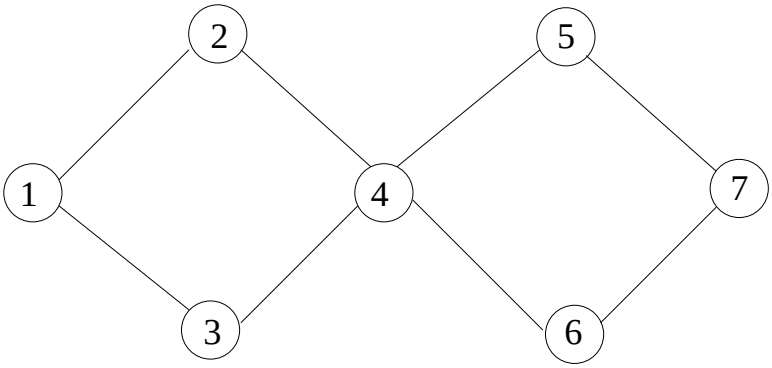

Figure 3.1: A network example

Table 3.1: The paths of two alternative solutions

Solution 1	Solution 2
1-2-4-5-7	1-2-4-6-7
1-3-4-6-7	1-3-4-5-7

The algorithm to define paths to one ingress-egress pair is as follows: We have the original topology T, which consists of the set of directed links. Because one ingress-egress pair uses only part of the whole topology, we define a new topology $T'_{(i,j)}$, which consists of the links for which $x_{(i,j),(m,n)}$ differs from zero. Topology $T'_{(i,j)}$ is loop-free, because if there were loops, they would be reduced and the original allocation would not be optimal. Let $L'_{(i,j)}$ be the number of

links in topology $T'_{(i,j)}$. We search all paths to one ingress-egress pair (i, j) by a breadth-first-search algorithm, which is defined below.

1. Let $P_{(i,j)}$ be the empty set of paths.
2. Find all links (i, k) from topology $T'_{(i,j)}$. Compose for each link (i, k) a subset $P'_{(i,j)}$, which consists of nodes i and k. Add $P'_{(i,j)}s$ to set $P_{(i,j)}$.
3. If the last component of subset $P'_{(i,j)}$ is egress node j, subset $P'_{(i,j)}$ is complete.
4. Consider one subset $P'_{(i,j)}$. Find all links (k, l), where k is the last node of $P'_{(i,j)}$, from topology T'. Compose for each link (k, l) a subset $P''_{(i,j)}$, which consists of the nodes of set $P'_{(i,j)}$ and node l. Now replace set $P'_{(i,j)}$ by sets $P''_{(i,j)}$ in set $P_{(i,j)}$. Repeat this to all subsets $P'_{(i,j)}$, which are not complete.
5. If there is some subset $P'_{(i,j)}$, which is not complete, go back to step 3, else stop the algorithm.

As an example of the breadth-first-search algorithm, we study how the path set $P_{(1,7)}$ is composed in the network presented in Figure 3.1. The topology of network is

$T = \{\{1,2\},\{1,3\},\{2,1\},\{2,4\},\{3,1\},\{3,4\},\{4,2\},\{4,3\},$
$\{4,5\},\{4,6\},\{5,4\},\{5,7\},\{6,4\},\{6,7\},\{7,5\},\{7,6\}\}$

and the flow allocation of ingress-egress pair $(1, 7)$ is

$x_{(1,7)} = \{50, 50, 0, 50, 0, 50, 0, 0, 50, 50, 0, 50, 0, 50, 0, 0\}$.

So the reduced topology $T'_{(1,7)}$ is

$T'_{(1,7)} = \{\{1,2\},\{1,3\},\{2,4\},\{3,4\},\{4,5\},\{4,6\},\{5,7\},\{6,7\}\}$.

The set $P_{(1,7)}$ in each iteration phase is presented in Table 3.2.

Table 3.2: The paths in each iteration phase

Phase	$P_{(1,7)}$
1.	$\{\}$
2.	$\{\{1,2\},\{1,3\}\}$
3.	$\{\{1,2,4\},\{1,3,4\}\}$
4.	$\{\{1,2,4,5\},\{1,2,4,6\},\{1,3,4,5\},\{1,3,4,6\}\}$
5.	$\{\{1,2,4,5,7\},\{1,2,4,6,7\},\{1,3,4,5,7\},\{1,3,4,6,7\}\}$

As a result of the algorithm, we have a set of possible paths $P_{(i,j)}$ for one ingress-egress pair. Let $K_{(i,j)}$ be the number of paths and $Q_{(i,j)} \in \mathbb{R}^{L'_{(i,j)} \times K_{(i,j)}}$ be the matrix, where

$$Q_{(i,j),(l,k)} = \begin{cases} 1, & \text{if path } k \text{ uses link } l \text{ of topology } T'_{(i,j)}, \\ 0, & \text{otherwise,} \end{cases} \tag{3.19}$$

for each ingress-egress pair (i, j). Let $Y_{(i,j),k}$ be the traffic allocation in path k. $Y_{(i,j),k}$s can be solved from matrix equation

$$Q_{(i,j)} Y^T_{(i,j)} = x'^T_{(i,j)}, \quad \text{for each } (i, j), \tag{3.20}$$

where $x'_{(i,j)} \in \mathbb{R}^{L'_{(i,j)}}$ is the flow vector for each ingress-egress pair (i, j). The system defined by matrix equation (3.20) can be over- or underdetermined, because the number of paths differs from the number of links. So it is useful to determine the pseudoinverse of $Q_{(i,j)}$ when solving the equations.

Finally, if element k of the solution vector $Y_{(i,j)}$ differs from zero, ingress-egress pair (i, j) uses path k, else not. So reducing these paths from path set $P_{(i,j)}$ we get the actual path set. The implementation of the algorithm to define paths is presented in Appendix B.

3.3.3 Non-linear programming formulation and solution

Now the set of paths is already solved with the linear programming and therefore the size of problem is reduced. The objective is to find an optimal flow allocation to these paths.

The additional notation introduced in [Ott01] is as follows:

$$\begin{aligned} &K = \textstyle\sum_{(i,j)} K_{(i,j)} = \text{total number of paths in the network,} \\ &P_{(i,j),k} = \text{path } k \text{ of node-pair } (i, j), \quad k = 1, ..., K_{(i,j)}, \\ &\phi_{(i,j),k} = \text{fraction of } d_{(i,j)} \text{ allocated to path } P_{(i,j),k}. \end{aligned}$$

The additional conditions are

$$\begin{aligned} &\phi_{(i,j),k} \geq 0, \\ &\sum_{k=1}^{K_{(i,j)}} \phi_{(i,j),k} = 1, \text{ for each } (i, j). \end{aligned} \tag{3.21}$$

The structure of paths is defined in [Ott01] by relations

$$\delta_{(m,n),(i,j),k} = \begin{cases} 1, & \text{if path } P_{(i,j),k} \text{ uses link } (m, n), \\ 0, & \text{otherwise.} \end{cases} \tag{3.22}$$

The link-loads are defined by the equation

$$X_{(m,n)} = \sum_{(i,j),k} \delta_{(m,n),(i,j),k} d_{(i,j)} \phi_{(i,j),k}. \tag{3.23}$$

Note that the traffic allocation $Y_{(i,j),k}$ presented in section 3.3.2 is a special case of traffic allocation $d_{(i,j)}\phi_{(i,j),k}$. The cost-function to minimize is the following:

$$\begin{aligned} &D(X) = \sum_{(m,n)} D_{(m,n)}(X_{(m,n)}) = \\ &\sum_{(m,n)} D_{(m,n)}\Big(\sum_{(i,j),k} \delta_{(m,n),(i,j),k} d_{(i,j)} \phi_{(i,j),k} \Big) = C(\phi), \end{aligned} \tag{3.24}$$

where each $D_(m,n)$ is an increasing convex function of its arguments. Our objective is to minimize the mean delay of the total network. So the optimization problem is as follows:

$$\begin{aligned} &\text{Minimize} \\ &\frac{1}{\Lambda} \sum_{(m,n)} \frac{\sum_{(i,j),k} \delta_{(m,n),(i,j),k} d_{(i,j)} \phi_{(i,j),k}}{b_{(m,n)} - \sum_{(i,j),k} \delta_{(m,n),(i,j),k} d_{(i,j)} \phi_{(i,j),k}}, \\ &\text{subject to the conditions} \\ &\sum_{(i,j),k} \delta_{(m,n),(i,j),k} d_{(i,j)} \phi_{(i,j),k} < b_{(m,n)}, \qquad \text{for each } (m,n), \\ &\phi_{(i,j),k} \geq 0, \qquad \text{for each } (i,j),k, \\ &\sum_{k=1}^{K_{(i,j)}} \phi_{(i,j),k} = 1, \qquad \text{for each } (i,j), \end{aligned} \tag{3.25}$$

where Λ is the total offered traffic of the network.

Paper [Ott01] notes that the sum $\sum_{(m,n)\in P_{(i,j),k}} D'_{(m,n)}(X_{(m,n)})$ is in the nature of a load-sensitive "path metric" for path $P_{(i,j),k}$. It can be calculated at each source node and transmitted to upstream neighbors. So the algorithm can use distributed calculation. In an optimal solution, all paths which have traffic allocated to them must have the same path-metric*, and other nodes must have path-metric, which is greater than or equal to path-metric* (compare with λ in equation (3.7)).

3.3.4 Three enhancements to the two-level procedure

In paper [Carp] Carpenter et al. present three enhancements to the routing algorithm introduced in [Ott01]. The first enhancement is the capability to select additional paths to the solution in advance, which increases the robustness of the routing. A method to create paths is simple. For each node-pair (i, j) with positive demand and only one path p, a shortest path q from i to j, which is disjoint from p is searched. This path q is added to the path set as a new LSP. The paper claims that it's adequate to have two paths per demand.

The second enhancement is the ability to choose only a single path to one ingress-egress node-pair. The optimization, which provides exactly one path to serve each ingress-egress pair, is an intractable integer linear program. So paper [Carp] suggests using paths only from LP-solutions when solving the integer program. The problem is that it is not guaranteed that the smaller integer program yields a good solution with respect to the original program.

The third enhancement is the capability to use admission control when it is impossible to carry all offered traffic. Paper [Carp] focuses on the admission control and formulates it as an LP-problem, by including the desired maximum link utilization as a parameter. The objective is to minimize the maximum loss-proportion among the demands.

3.4 Heuristic approach

In paper [Srid00] traffic flows between ingress and egress nodes are assumed to be known. The paper's two aspects are how to match traffic to the paths so as to achieve optimal performance and how often to update traffic allocation to the paths with respect to variations in traffic demands. 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.

The cost of achieving the optimal routing can be divided into two different parts, the first part is the classification cost in ingress node, which depends on the granularity and the second part is the forwarding cost. So the level of granularity remains an important criterion to keep the costs of the traffic aware routing low. Costs also grow with the frequency at which routers need to be adjusted when traffic demands change.

In paper [Srid00] prefix masks of different lengths are used for the traffic aggregation decision. So the used prefix lengths define the level of granularity, example length of zero refers to the coarsest granularity level, i.e. there is only a single path per ingress-egress pair.

The heuristics to study the granularity is very simple. Depending on the level of

granularity, traffic demands from ingress to egress node are divided into streams (e.g. using some hashing function). Streams are added one at a time onto route which minimizes delay. The streams can be routed in ascending, descending or random order in terms of their traffic intensity for example.

Let $g \in \mathbb{N}$ be the level of granularity. Traffic demands from ingress node i to egress node j are given by a matrix $[d_{(i,j)}]$. We compose a list D consisting of ingress-egress pairs and their traffic intensities. The elements in the list D are sorted in (i) ascending order in terms of their traffic demand, (ii) descending order in terms of their traffic demand, (iii) descending order in terms of their shortest mean delay from ingress node to egress node calculated in the empty network.

We route each element of the list D sequentially one at the time. The algorithm to route element l is as follows:

1. Calculate link costs $c^{(l)}_{(m,n)}$ using the mean delay of an $M/M/1$-queue:

$$c^{(l)}_{(m,n)} = \frac{1}{b_{(m,n)} - (a^{(l)}_{(m,n)} + \dfrac{d^{(l)}}{g})}, \tag{3.26}$$

   where $[a^{(l)}_{(m,n)}]$ is traffic matrix that contains all traffic that is already routed to link (m,n) (in the first phase $a^{(1)}_{(m,n)} = 0$ for each (m,n)).

2. Calculate the shortest path using Dijkstra's algorithm 1. Use $c^{(l)}_{(m,n)}$ as link cost.

3. Add traffic flow $d^{(l)}/g$ to these components of matrix $[a^{(l)}_{(m,n)}]$, which are included in the solution of Dijkstra's algorithm.

4. Repeat phases 1-3 g times, so that all traffic of the ingress-egress pair is routed.

Algorithm 1 (Dijkstra) Let i be the source node, j the destination node, P the set of nodes, $C_{(i,j)}$ the total length of shortest path from node i to node j. Initially, $P = \{j\}$, $C_{(j,j)} = 0$ and $C_{(i,j)} = c^{(l)}_{(i,j)}$ for $i \neq j$.

1. Find the next closest node $k \notin P$ such that $C_{(k,j)} = \min\limits_{i \notin P} C_{(i,j)}$. Set $P := P \cup \{k\}$. If P contains ingress node i, the algorithm is complete and you can stop the algorithm.
2. Update labels: For all $i \notin P$ set $C_{(i,j)} := \min\left[C_{(i,j)}, c^{(l)}_{(i,k)} + C_{(k,j)}\right]$. Go to step 1.

The procedure above is repeated to each ingress-egress pair. Finally, the mean delay in the whole network can be calculated using

$$C(\gamma) = \frac{S}{\Lambda} \sum_{(m,n)} \frac{a_{(m,n)}}{b_{(m,n)} - a_{(m,n)}}, \tag{3.27}$$

where S is the average packet size and Λ is the total offered traffic.

The computation of Dijkstra's algorithm needs $N - 1$ iterations, where N is the number of nodes in the network [Bert92]. So the computation is $O(N^2)$ in the worst case. The total implementation of the heuristics needs the level of granularity times the number of ingress-egress pairs calculations of Dijkstra's algorithm. The implementation of heuristics is presented in Appendix B.

3.5 Some other proposed routing methods

3.5.1 Static versus traffic-aware routing

Paper [And01] attempts to quantify the performance of static versus traffic-aware routing. The static routing policies examined are the shortest path routing based on the hop-count and the shortest path routing based on the inverse link bandwidth. The two traffic-aware policies examined are minimizing the maximum link utilization and minimizing the mean queuing delay. Further, paper [And01] develops two novel traffic-aware routing algorithms.

Shortest path and weighted shortest path routing algorithms are the primary routing algorithms in use today. In OSPF (open shortest path first) link weights are set by default to one, so the shortest path is achieved by minimizing the number of hops. OSPF link weights can also be determined in proportion to the reciprocal of the link capacity (referred to as autocalculated weights). So the use of OSPF autocalculated weights can provide the weighted shortest path solution.

The two novel algorithms of paper [And01] take traffic requirements into account but do not explicitly evaluate the non-linear function. The first own algorithm is based on the algorithm that minimizes the maximum link load. However, because the path lengths are ignored in this algorithm, the routing can use paths that are much longer than necessary. Therefore, the paper presents a modified multicommodity flow optimization. This algorithm tries to both limit the worst-case link utilization and minimize path lengths to the extent possible. The algorithm is as follows: (1) Compute the minimum worst-case utilization λ^{-1}, (2) increase λ by a small amount, (3) recompute a new routing solution that minimizes the total path length subject to the constraint given by the adjusted λ. This "bi-optimization" optimization is simpler to implement using well-known linear optimization techniques rather than Gallager's algorithm.

The primary performance metric of paper [And01] is the sum of link delays.

The paper's second own algorithm uses Gallager's routing tables to divide traffic equally among a limited number of outgoing links. At each node and for each destination, set of outgoing links are selected so that the smallest redistribution compared to the optimal solution is required. The paper decides to limit the number of outgoing links to at most three. This algorithm is called "Rounded Gallager Tables".

As a result, [And01] conclude that performance of traffic aware algorithms, "bi-optimization" and "Rounded Gallager Tables", are close to optimal when compared to the shortest path algorithms. The shortest path algorithms reach their breakpoints at the lower traffic load than the traffic-aware algorithms.

3.5.2 Traffic Engineering using DiffServ model

Paper [Vut00] investigates the potential of minimum-delay routing. The paper uses minimum delay routing formulated by Bertsekas and Gallager [Bert92] as a basis for its own traffic engineering technique. However, the complexity in the MPLS implementation can be overcome, if the flows are set up using routing parameters obtained by optimal minimum-delay routing.

Two types of traffic are assumed: one tolerates out-of-order packet delivery (e.g. UDP) while the other does not (e.g. TCP). These types will be referred to as UDP and TCP traffic. The granularity of allocation is at the flow level for TCP traffic and at the packet level for UDP-traffic.

To forward TCP traffic the paper uses an architecture similar to the Diffserv model. At the ingress node, a key is generated randomly for each TCP connection. When a packet arrives, this key is inserted into the packet. The key is used to hash packet into the next-hop. TCP traffic cannot be totally controlled, so actual traffic forwarded can deviate significantly from amounts specified by routing parameters obtained by optimal solution. So UDP traffic can be forwarded to the successors, which receives too little traffic compared to routing parameters.

3.5.3 MATE

MPLS Adaptive Traffic Engineering (MATE) [Wid98] assumes that there are M explicit LSPs established between each ingress and egress node (M is typically between 2 and 5). These M paths are calculated by some algorithm to be "best". When these paths are set up, the ingress node tries to distribute traffic across the paths so that the traffic loads are balanced and congestion minimized.

The algorithm can be divided into four phases. First the congestion of each LSP is measured. The measurement can be a function of delay derivative and packet loss, if packet loss is remarkable. Packet delay changes between ingress and egress

nodes are calculated using probe packet and packet loss probability is calculated using a bunch of probe packets. In the second phase, the algorithm tries to equalize the congestion measure for each LSP. In the third phase, the algorithm monitors each LSP. If an appreciable change of the network state is detected, the algorithm goes to the fourth phase, otherwise it goes back to the second phase. In the fourth phase congestion measures are adjusted. Then the algorithm goes back to the second phase.

Chapter 4

Quality of Service

4.1 Introduction

There are many definitions for the concept of Quality of Service (QoS) [Fer98]. It can be defined as the ability to provide resource assurance and service differentiation in a network [Wan01].

In today's Internet services are provided only on the best-effort basis. This means that there are no guarantees for any certain level of packet loss and transmission delay etc [Gué99]. The demand for various services exists both among companies that make business using Internet and among private users. So there is a strong possibility that Internet will contain several service classes. Traffic can be divided into traffic classes like gold service, silver service and bronze service so that the usage of the gold service is most expensive but guarantees better quality compared to the silver and bronze services [Xia99].

There is plenty of discussion about the mechanisms to provide quality of service nowadays. One opinion is that optical fibers and wavelength-division multiplexing (WDM) enable sufficient bandwidth to all application and QoS is therefore guaranteed. Another opinion assumes that resources are everlastingly or for very long time limited, so some mechanism to provide QoS is needed. This opinion is the basis for our further examination.

Internet Engineering Task Force (IETF) has introduced many QoS models and mechanisms. Two of these, Integrated Services [RFC1633] and Differentiated Services [RFC2475] are presented more detailed in sections 4.2 and 4.3. Also scheduling mechanisms like Weighted Fair Queueing are explained in section 4.4. The function of the scheduling mechanisms is to control how network's resources are divided to each connection. The scheduler is a crucial part of the Differentiated Services architecture. Last two sections concentrate on proposed practical mechanisms to provide quality of service. We introduce QoS routing scheme and

its implementation approaches in section 4.5. Section 4.6 focuses on solutions to obtain Differentiated Services using MPLS.

4.2 Integrated Services

The term Integrated Services (IS) refers to an Internet service model that consists of best-effort service, real-time service and controlled link sharing [RFC1633]. Two types of service are introduced to meet the requirements of real-time traffic: guaranteed and predictive service. These services are integrated using controlled link-sharing. IS supports both unicast and multicast transmission.

The IS model assumes that, to achieve the requirements of applications, the resources of the network must be controlled. This is based on resource reservation and admission control. The resource reservation is done using Resource Reservation Protocol (RSVP) the first version of which was specified in [RFC2205]. A host uses RSVP to request a specific level of quality of service for some subset of packets in a particular session. RSVP operates together with unicast and multicast routing protocols. It uses a local routing database to obtain routes and request resources in one direction at a time.

A reservation request includes a "flowspec" and a "filter spec". The quality of service, i.e resource quantity, is defined by the flowspec. The filter spec together with a session specification defines which packets are designed to receive the quality of service defined by the flowspec.

The implementation of IS is divided into four parts according to [Xia99], which are signalling protocol (e.g. RSVP), admission control routine, the classifier and the packet scheduler. The signalling protocol reserves resources before data is transmitted. The admission control routine determines if the reservation request made by the signalling protocol can be allowed. The classifier classifies packets using multifield (MF) classification and forwards packets to the different queues depending on the classification. The packet scheduler then schedules packets so that the desired QoS is obtained.

Integrated Services is one step towards Quality of Service in the Internet. However, there are some difficulties in the IS model [Xia99]. The architecture does not scale well, because the amount of state information increases in proportion to the number of flows and this causes a huge overhead in routers. The technology in the routers have to be of a high level to handle resource and admission control.

4.3 Differentiated Services

IETF has introduced Differentiated Services (DS) as a mechanism to classify IP traffic to service classes. The purpose of Differentiated Services is not to provide guaranteed services, but packets in the higher service class have a higher probability to get through or they have a smaller mean delay. The architecture of Differentiated Services is defined in [RFC2475].

4.3.1 The key elements

IPv4 packet header includes the Type of Service (TOS) field, which contains essential information to perform differentiation in the network. In IPv6 packet the corresponding field is the Traffic Class field. Because information is carried in the packet header, Differentiated Services does not need any signalling protocol (e.g RSVP) to control the process. That is why the state information increases proportionally to the number of service classes, not proportionally to the number of flows, like in Integrated Services.

A customer has to have a service level agreement (SLA) with its Internet service provider (ISP) in order to get differentiated services. SLA defines the desired level of performance and features of service statically or dynamically. Static SLA's are made for a regular time interval. Dynamic SLA is based on using signalling protocol (e.g RSVP) to request service. Service Level Specification (SLS) is a subset of SLA and defines the technical specifications of the service.

The main elements of Differentiated Services are packet classifiers, traffic conditioners (see Figure 4.1) and per-hop forwarding behaviors (PHB).

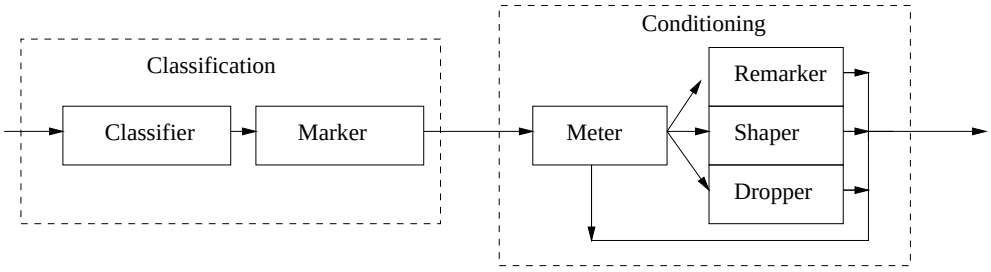

Figure 4.1: Classification and conditioning

4.3.2 Packet classification

A DS node is a node that is capable to support Differentiated Services functions and behaviors. A DS domain is a set of DS nodes, which works according to a common service provisioning policy. Packet classification is done by classifiers

at the edge of a DS domain and its purpose is to identify packets which belong to a specific traffic stream and receive differentiated services. Classifiers operate according to precise rules defined by a Traffic Conditioning Agreement (TCA).

There are two types of classifiers. The function of Behavior Aggregate (BA) classifier is based on the DS codepoint, whereas Multi-Field (MF) classifier uses several header fields, such as source address and destination address, in the classification process. The information that is used to classify packets should be authenticated.

4.3.3 Traffic conditioners

Traffic conditioners consist of one or more of the following elements: meter, marker, shaper and dropper. These elements build up the most important part of DS network. The purpose of conditioners is to apply conditional functions on the categorized packets using Traffic Conditioning Specification (TCS).

The traffic meter is a device that meters temporal properties of the selected traffic stream and compares these properties to the traffic profile defined in a TCA. The meter transmits state information to other conditioning functions to activate some specific action.

The marker is a devise that adds a packet to a particular DS behavior by setting packet's DS field to the particular codepoint. Two forms of activity of the markers are introduced in [RFC2475]. In the first form a marker marks all packets that traverse through it to a single codepoint, whereas in the second form a marker marks packets according to the state of a meter.

The shaper is a device that delays packets of a traffic stream using a finite-length buffer, and its idea is to get the traffic stream into compliance with a traffic profile.

The dropper is a device that also tries to bring a traffic stream into compliance with a traffic profile by dropping some or all packets.

4.3.4 Per-Hop Behaviors

A Per-Hop Behavior (PHB) describes the externally observable forwarding behavior of a particular DS node applied to a group of packets which have the same DS codepoint and the same link direction (also known as DS behavior aggregate). Each service class uses some PHB. PHBs can also be defined as relations in resource priorities or observable traffic characteristic (e.g. delay and packet loss).

The implementation of PHBs is done in nodes by using buffer management and

packet scheduling. The definition of a PHB includes service provisioning specifications, not implementation characteristics. So the implementation of a PHB group can vary. If more than one PHB groups are implemented in a node, the resource allocation between groups should be able to be concluded.

The three proposed PHBs are Default (DE) PHB, Expedited Forwarding (EF) PHB and Assured Forwarding (AF) PHB [Andr99]. They are introduced below.

DE PHB offers forwarding on the best-effort basis and is used widely in the Internet today. There are no particular rules when packets are forwarded in network. The target is to deliver packets as many as possible without any guarantees.

EF PHB is aimed to a particular differentiated service aggregate for which the service rate in each DS node must equal or exceed a configurable rate. EF is a high priority PHB and often used for control traffic. The details of EF are described in [RFC2598].

The purpose of AF PHB group is to provide different levels of forwarding assurances for IP packets that are received from a customer DS domain. AF PHB group is defined to have four AF classes. Each AF class receives a certain amount of forwarding resources. IP packets can be marked with one of three possible drop precedence values. If there is congestion in the network, the packets with a high drop precedence value are dropped and lost rather than packets with a low drop precedence value. For more information, see [RFC2597].

4.4 Weighted Fair Queueing (WFQ)

The selection of a packet service discipline at the switching element is one major aspect when trying to provide guaranteed bandwidth and Differentiated Services. Without special control, packets from different connections can interact with each other and this interaction has an effect on each connection's experienced service. A service discipline controls how packets from different connections are delivered and how they interact with each other.

There are two types of service disciplines, work-conserving and nonwork-conserving service disciplines. With a nonwork-conserving discipline, only the packets that are eligible could be forwarded. With a work-conserving discipline, if there are any packets in the queue, a server is never idle. One example of work-conserving service disciplines is Weighted Fair Queueing (WFQ) [Zha95, Ben96, Sem], which is next briefly explained.

The idea of WFQ is to provide a fair bandwidth distribution to queues with different bandwidth requirements. The method of WFQ is to assign a weight to each queue and then provide a percentage of output port bandwidth to queues according to these weights (see Figure 4.2). WFQ supports also the fair band-

width allocation to variable-length packets, but this increases the complexity of the queue scheduling algorithm significantly.

WFQ and it's variation WF^2Q are policies that try to approximate Fluid Fair Queueing (FFQ) policy (also known as Generalized Processor Sharing (GPS)). With FFQ, the server serves the first packet of each non-empty queue during the time interval. Service shares can be different among different connections. However, FFQ is not practical, because the server is assumed to be able to serve all connections simultaneously and traffic is assumed to be infinitely divisible.

When approximating FFQ by WFQ, a finish time (actual transmission time) to each packet is calculated and assigned based on the bandwidth of the output port, number of active queues and relative weights of each queue. Then WFQ selects a packet with the smallest finish time and forwards it. WF^2Q differs from WFQ by taking also the starting time into account. In WF^2Q, a packet with the shortest finish time is selected. A restriction is that this packet should already have started to receive service in the corresponding FFQ system.

The advantage of WFQ is that it ensures the desired portion of the output port bandwidth to each service class in spite of the behavior of other classes. So the delays of each class are limited. On the other hand, implementations of WFQ are in software level, which can slow down the interfaces of the network.

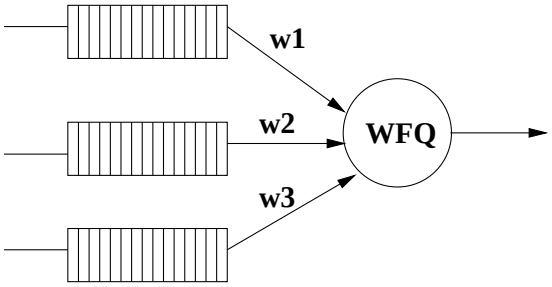

Figure 4.2: Weighted Fair Queueing

4.5 QoS routing mechanisms

QoS-based routing is a routing mechanism, where paths for a particular flow are selected based on the information of available resources in the network and QoS requirements of flows [RFC2386]. The three major elements of QoS-based routing according to [RFC2676] are 1) acquiring the required information and determining a feasible path based on QoS requirements of a given request, 2) accommodation of a new request by establishing the path selected and 3) the maintenance of the selected path.

QoS routing can be viewed as a missing piece of the puzzle when achieving actual quality of service [Fer98]. The area includes many problems like deciding whether

the quality is provided on the link basis or on the end-to-end basis.

OSPF extensions

The extension of Open Shortest Path First (OSPF) algorithm to support QoS routing is introduced in [RFC2676]. The additions can be divided into three parts: 1) The metrics required to support QoS, 2) the extensions in OSPF to advertise the updates of QoS metrics over network and 3) the modifications in path selection algorithms.

The network tries to select the "cheapest" paths of all feasible paths. The most important metrics that have an influence on the selection are available bandwidth of the links, link propagation delay and hop-count. These metrics have to be advertised over the network to each router. One choice is to use periodic updates, where the tolerance of the change in the traffic load determines the update period. Another and preferable choice is to trigger link state advertisements only when the values of metrics change remarkably.

Path selection includes two aspects, the path selection algorithm itself and the decision when path selection is invoked. The purpose of the algorithm is to select paths which use the minimum number of hops among the links that can offer the requested bandwidth. So the load is balanced and the likelihood that the required bandwidth is available is maximized. The path selection algorithm can be invoked on-demand basis or paths can be selected in advance. [RFC2676] focuses on the pre-calculation of paths that is done after N updates in the metrics.

The objective is to restrict changes in OSPF code to minimum and keep interactions between original and extensions code small.

Inter-domain solutions

The mechanism described above can be regarded as an intra-domain solution, which may be used in autonomous systems (AS). However, there exists a need for inter-domain solution. The inter-domain model described in [RFC2386] includes the set of hierarchically organized ASs. The information exchange between domains has two challenges, how to determine the level of QoS that is supported along the path to a particular destination and how to express the routing policies like monetary cost as a function of flow parameters.

In the inter-domain model of [RFC2386], the border nodes calculate first the level of locally available QoS, "AS metrics", in order to be able to advertise the routes to the destinations, which can be located in different ASs. The computation of the paths is based on the knowledge of the traffic that is expected to arrive from neighboring ASs and the corresponding QoS needs. In the inter-domain QoS routing, the essential requirement is scalability.

Paper [Nor02] studies also the routing of QoS sensitive traffic across ASs. The most common inter-domain routing protocol, BGP, is not suitable for QoS routing, due to the slow link recovery. Also there is short of uniform metric or a policy to find globally optimal end-to-end routes. Paper [Nor02] makes an attempt to overcome these problems by developing two new inter-domain QoS routing algorithms. The first of algorithms is based on the conventional shortest-path approach, where the cost metric consists of both inherent cost of link and the amount of available bandwidth. The basic idea of second algorithm is to search several paths parallel, which reduces the need for regular routing updates.

4.6 Differentiated Services over MPLS

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

One proposal to achieve QoS is a Provider Architecture for Differentiated Services and Traffic Engineering (PASTE) [RFC2430]. PASTE utilizes MPLS and RSVP to provide scalable traffic management architecture. Traffic is divided into three classes: Best Effort, Priority and Network Control. The basic concept of PASTE is a traffic trunk that enables hierarchical aggregation of traffic. A traffic trunk consists of traffic flows that are aggregated according to their classes and placed inside an LSP. Trunks can also be aggregated and merged in order to minimize the number of trunks and improve scalability. Traffic trunks are constituted and maintained using RSVP.

PASTE uses explicit routes of the RSVP signalling structure when allowing ISPs to perform traffic engineering. Traffic handling operations can be done now on a trunk-basis. As a result, PASTE provides traffic engineering mechanisms to ISP, which should scale well in a large network.

Another possibility to achieve Quality of Service is an infrastructure introduced in [RFC3270]. The structure specifies a solution that supports Behavior Aggregates (BA) of Differentiated Services in an MPLS domain. The solution presented in [RFC3270] includes two types of LSPs, EXP-Inferred-PSC LSP and EXP-Inferred-PSC LSP.

EXP-Inferred-PSC LSP (E-LSPs) is an LSP that can support one or more Ordered Aggregates (OA). Ordered Aggregate is a set of BAs that share an ordering constraint. The sharing of an ordering constraint means that the packets with in an OA need to depart in the same order than they have arrived. An LSR uses the EXP field of MPLS Shim header when determining which PHB should be applied to the packet. PHB Scheduling Class (PSC) consists of PHBs that are

applied to BAs of a given OA. With E-LSPs, PSC is specified by the EXP field of the packet.

EXP-Inferred-PSC LSP (L-LSP) is an LSP that can support only a single FEC-OA pair. PSC is signaled explicitly when a label is established and LSR can conclude the PSC of a labelled packet only from the label value.

An MPLS DS domain of the model in [RFC3270] consists of a combination of E-LSPs and L-LSPs, which is selected by the network administrator from a set of permitted combinations. The network administrator defines also how BAs are forwarded over this combination of LSPs in order to fulfil Quality of Service requirements. In a given FEC, several LSPs can carry the same OA, but in order to attain ordering constraints, all packets in a given micro-flow must use the same LSP. Respectively, LSPs must also support all the BAs of a given OA. LSPs can be set up with or without bandwidth reservation. However, the operation of the forwarding element, which includes scheduling and drop policy, is defined by the Diff-Serv PHB. Bandwidth reservation is done using an extended RSVP model.

Internet Draft [Fau02] introduces Service Provider requirements that enable Diff-Serv aware MPLS Traffic Engineering (DS-TE). In the DS-TE model traffic engineering is done at a per-class level instead of the aggregate level. Traffic of a particular class can be mapped to separate LSPs so that the resources of a network are utilized. The draft focuses on three special cases, a network with limitations in resources, a network that have notable amounts delay-sensitive traffic and a network where significant amounts of traffic between classes are not uniform.

Chapter 5

Routing methods for differentiating quality of service

5.1 Introduction

As mentioned in chapter 4, there exists a strong demand to provide guaranteed or differentiated services in the Internet. Many architectures have been developed and implemented in recent years, only a part of them are introduced in chapter 4. On the other hand, MPLS as a tag switched technology provides many possibilities to enhance QoS through internet traffic engineering.

Our goal is to develop routing methods that optimize differentiation of experienced service of different classes in terms of the mean delay. We use routing methods that are introduced in chapter 3 as a starting point in the further development. These methods make an attempt to balance load in the network and thus achieve minimum mean delay.

The routing methods to be introduced are divided into two types. The first type tries to optimize only flow allocation so that differentiation is achieved. The traffic class that should achieve a small mean delay compared to the other classes is routed along the path that is not congested, for example. The second type of methods makes use of WFQ-weights. Each node has a WFQ-weight that determines, which proportion of the bandwidth is assigned to each service class. Also these methods optimize the flow allocation by minimizing the mean delay.

The number of free variables grows in proportion to the size of network and the optimization becomes intractable. So one goal is to find near optimal solutions using methods that reduce the size of the problem. One example is the use of both linear and non-linear optimization introduced in section 3.3.

The differentiation of the mean delay in the whole network does not mean that the mean delays of pairs that have the same ingress and egress nodes are different

between different classes. One approach is to fix the mean delays at every link so that the differentiation is guaranteed at the ingress-egress pair level also.

5.2 Methods to achieve differentiation in mean delay using routing

5.2.1 Optimization using cost weights

We consider the situation where the total traffic is divided into traffic classes, into the gold, silver and bronze classes, for example. The gold class has the highest priority and bronze class the lowest priority. Each traffic class l has its own traffic matrix $[d_{l,(i,j)}]$, where i is the ingress node and j is the egress node. $R_{l,(i,j)} \in \mathbb{R}^N$ is an array for each class l and ingress-egress pair (i,j), where $R_{l,(i,j),k} = d_{l,(i,j)}$, if k is the ingress node, $R_{l,(i,j),k} = -d_{l,(i,j)}$, if k is the egress node and $R_{l,(i,j),k} = 0$ otherwise. The total traffic offered by class l is denoted by Λ_l. Let $x_{l,(i,j),(m,n)}$ be the allocated traffic of ingress-egress pair (i,j) of class l on link (m,n). So the total traffic of class l on link (m,n) is

$$X_{l,(m,n)} = \sum_{(i,j)} x_{l,(i,j),(m,n)}, \quad \text{for each } l,(m,n). \tag{5.1}$$

Let w_l be the cost weight associated to traffic class l. Additional notation used is the same as in section 3.3.

The purpose is to divide traffic into paths so that classes with higher priority, like gold class, achieve smaller mean delay than other classes. So the cost of delay is highest in the gold class and so on. The optimization problem, where we minimize the sum of the weighted mean delays of the classes is as follows:

Minimize

$$\sum_l w_l E[D_l] = \sum_{(m,n)} \frac{\sum_l \frac{w_l X_{l,(m,n)}}{\Lambda_l}}{b_{(m,n)} - \sum_l X_{l,(m,n)}}, \tag{5.2}$$

subject to the constraints

$$\sum_l X_{l,(m,n)} < b_{(m,n)}, \qquad \text{for each } (m,n),$$
$$Ax^T_{l,(i,j)} = R^T_{l,(i,j)}, \qquad \text{for each } l,(i,j).$$

When the cost weights of different classes differ enough, the optimization function tries to minimize the mean delays of high priority classes at the expense of lower

priority classes. The high priority class can be routed through the shortest path and the other classes through some other path in order to provide enough capacity to the high priority class. As a result, the routing obtained by the optimization function above differs from the routing obtained by the load balancing, because in the load balancing approach the delays in the nodes are balanced to be equal and the differentiation could occur only if the paths are of different length.

In practice, the optimization area should be more constrained than is presented in the first constraint of (5.2) in order to avoid situations where the denominator of the optimization function is near to zero and the value of the optimization function is very large. This can be done by multiplying bandwidth $b_{(m,n)}$ by a multiplier that is a little less than one.

5.2.2 Optimization with a fixed mean delay ratio

Now we fix the ratio of the mean delays of various classes to some value in order to differentiate classes. For example, in the case of two classes (gold and silver), the ratio of the mean delays between the silver and the gold class could be fixed to parameter q. After that the optimization can be done by minimizing the mean delay of either class. Also the sum of the mean delays can be minimized. The optimization problem with the constraint that fixes the ratio of the mean delays is presented in (5.3).

Minimize

$$E[D_{l_1}] = \frac{1}{\Lambda_{l_1}} \sum_{(m,n)} \frac{X_{l_1,(m,n)}}{b_{(m,n)} - \sum_l X_{l,(m,n)}},$$

subject to the constraints

$$\begin{aligned} \frac{E[D_{l_2}]}{E[D_{l_1}]} &= \frac{\frac{1}{\Lambda_{l_2}} \sum_{(m,n)} \frac{w_{l_2} X_{l_2,(m,n)}}{b_{(m,n)} - \sum_l X_{l,(m,n)}}}{\frac{1}{\Lambda_{l_1}} \sum_{(m,n)} \frac{w_{l_1} X_{l_1,(m,n)}}{b_{(m,n)} - \sum_l X_{l,(m,n)}}} = q, \\ \sum_l X_{l,(m,n)} &< b_{(m,n)}, \qquad \text{for each } (m,n), \\ A x^T_{l,(i,j)} &= R^T_{l,(i,j)}, \qquad \text{for each } l, (i,j). \end{aligned} \tag{5.3}$$

Compared to the optimization in the previous section, this optimization function does not include cost weights and the actual ratio of the mean delays is known before the optimization. It is useful to constraint the ratio of the mean delays to some small domain in order to make the optimization procedure easier.

5.2.3 Heuristic approach

There exists a demand to provide also simple routing methods that can be implemented without heavy optimization using distributed calculation. The approach that routes traffic to the network near optimally but still obtains the differentiation in terms of mean delay tries to adapt the heuristic approach presented in section 3.4.

The heuristic approach in the case of two classes (gold and silver) is as follows:

1. The gold class is routed using heuristics introduced in 3.4.
2. The allocated traffic of the gold class (denoted by $a_{(m,n)}$) is multiplied by $1 + \Delta$.
3. The silver class is routed using heuristics introduced in 3.4.

The idea of heuristics is that the links that are used by the gold class look more congested than they actually are. So the routing scheme tries to balance load by routing the silver class by some other way. That is, the artificial congestion forces the silver class to avoid links used by the gold class and therefore the gold class should achieve more bandwidth. The choice of parameter Δ depends on how much there is traffic offered compared to the capacity of the network.

5.3 Methods to achieve differentiation in mean delay using WFQ-weights

5.3.1 The cost weights included in the optimization function

The possibilities to provide differentiated services using routing only are limited. To achieve certain ratio of mean delays may lead up to incompetent routing because the low priority class is routed along long and congested paths in order to artificially obtain the desired ratio of the mean delay.

Weighted Fair Queueing as a packet scheduling mechanism (introduced in section 4.4) divides bandwidth among the different queues. Each queue achieves guaranteed bandwidth, which depends on the WFQ-weighs. We try to find optimal routing that differentiates the quality of service by including the WFQ-weighs to the optimization function as free parameters.

Because WFQ-scheduling is a work-conserving discipline, the bandwidth that is guaranteed for a class in our model can be viewed as the lower bound. Actually,

the bandwidth can be much greater due to the statistical behavior of different traffic sources.

Straightforward optimization

The bandwidth of each link is shared according to WFQ-weights. We approximate the behavior of WFQ-scheduling as follows: Let $\gamma_{l,(i,j)}$ be the WFQ-weight that determines the proportion of total bandwidth that is given to class l. There exist the following constraints:

$$\begin{aligned} & b_{l,(m,n)} = \gamma_{l,(m,n)} b_{(m,n)}, && \text{for each } l,(m,n), \\ & \sum_{l} \gamma_{l,(m,n)} = 1, && \text{for each } (m,n). \end{aligned} \tag{5.4}$$

The optimization problem where the sum of weighted mean delays is minimized is presented in (5.5) and is referred to as "straightforward". The differentiation between classes is obtained by using cost weights as in (5.2). The gold class should have the greatest cost weight and so on.

Minimize

$$\sum_{l} w_l E[D_l] = \sum_{l} \frac{w_l}{\Lambda_l} \sum_{(m,n)} \frac{X_{l,(m,n)}}{\gamma_{l,(m,n)} b_{(m,n)} - X_{l,(m,n)}},$$

subject to the constraints

$$\begin{aligned} & X_{l,(m,n)} < \gamma_{l,(m,n)} b_{(m,n)}, && \text{for each } l,(m,n), \\ & A x^T_{l,(i,j)} = R^T_{l,(i,j)}, && \text{for each } l,(i,j), \\ & 0 < \gamma_{l,(m,n)} < 1, && \text{for each } l,(m,n), \\ & \sum_{l} \gamma_{l,(i,j)} = 1, && \text{for each } (i,j). \end{aligned} \tag{5.5}$$

Two-step algorithms

The optimization problem presented in (5.5) is quite heavy and time-consuming. If the allocated traffic exceeds the bandwidth, the value of the objective function goes up to infinity. The constraints that prevent the allocated traffic to exceed the bandwidth of a particular link have an effect on the value of objective function. For example, in the situation where some class does not use a particular link, the WFQ-weight of such class should be zero in order to provide maximum bandwidth to other class. Now the value of denominator of the objective function is zero and the value of the objective function is infinity.

We introduce near optimal algorithms that make the size of the problem smaller and the calculation easier. The first two algorithms allocate traffic into the network and after that optimize WFQ-weights. Both algorithms are as follows:

1. Allocate the traffic into the network without WFQ-weights so that the sum of mean delays is minimized. The formulation of the optimization algorithm is presented in section 5.2.1.

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

3. Determine WFQ-weights using optimization problem (5.5). Now the number of free variables equals to the number of WFQ-weights that is the number of links in the network multiplied by the number of classes minus one.

The cost weights of the optimization function are selected twice. The difference between the first and second algorithms is that in the first two-step algorithm the cost weights of the first step are equal to one (referred to as "two-step, version 1") and in the second two-step algorithm the cost weights of the first and second steps are equal (referred to as "two-step, version 2").

The first two-step algorithm makes use of only WFQ-scheduling when trying to differentiate classes. It is reasonable since the flow allocation is optimal and artificial differences in mean delays do not appear in the first step. The second two-step algorithm instead utilizes both routing and WFQ-scheduling when differentiating classes and can therefore be closer to the optimal.

The third approximative algorithm makes use of the two-step algorithm presented in section 3.3. The paths are first calculated using the linear optimization that minimizes the maximum link load. Then the traffic is allocated to the paths using the non-linear optimization. The algorithm (referred to as "QoS-LP-NLP") is as follows:

1. Calculate the traffic allocation that minimizes the maximum link load using algorithm presented in section 3.3.1. Consider the traffic of all classes as one aggregate.

2. Calculate the paths corresponding to the traffic allocation of the previous step using the algorithm presented in section 3.3.2.

3. Allocate traffic to the paths obtained in step 2 using optimization function (5.6), where $K_{(i,j)}$ is the number of paths of ingress-egress pair (i,j), $P_{(i,j),k}$ is path k of node-pair (i,j), $\delta_{(m,n),(i,j),k}$ is one if path $P_{(i,j),k}$ uses link (m,n) and zero otherwise and $\phi_{l,(i,j),k}$ is fraction of traffic $d_{l,(i,j)}$ that is allocated to path $P_{(i,j),k}$.

Minimize

$$\sum_l w_l E[D_l]$$

$$= \sum_l \frac{w_l}{\Lambda_l} \sum_{(m,n)} \frac{\sum_{(i,j),k} \delta_{(m,n),(i,j),k} d_{l,(i,j)} \phi_{l,(i,j),k}}{\gamma_{l,(m,n)} b_{(m,n)} - \sum_{(i,j),k} \delta_{(m,n),(i,j),k} d_{l,(i,j)} \phi_{l,(i,j),k}},$$

subject to the constraints

$$\begin{aligned}
&\phi_{l,(i,j),k} \geq 0, && \text{for each } l,(i,j),k,\\
&\sum_{k=1}^{K_{(i,j)}} \phi_{l,(i,j),k} = 1, && \text{for each } l,(i,j),\\
&\sum_{(i,j),k} \delta_{(m,n),(i,j),k} d_{l,(i,j)} \phi_{l,(i,j),k} < \gamma_{l,(m,n)} b_{(m,n)}, && \text{for each } l,(m,n),\\
&0 < \gamma_{l,(m,n)} < 1, && \text{for each } l,(m,n),\\
&\sum_l \gamma_{l,(i,j)} = 1, && \text{for each } (i,j).
\end{aligned} \tag{5.6}$$

5.3.2 Optimization with fixed link delay ratios

In the routing with WFQ-weights, if the ratio of mean delays is fixed like in the algorithm presented in section 5.2.2, the optimization problem is intractable. One possibility is to fix the mean delay ratios at the link level. If the lengths of paths of different classes do not differ significantly, this approach should result in the same mean delay ratio in the whole network.

We consider the case with two classes, gold and silver. We fix the ratio of link delays to parameter q, that is

$$\frac{E[D_{l_2,(m,n))}]}{E[D_{l_1,(m,n)}]} = \frac{\gamma_{l_1,(m,n)} b_{(m,n)} - X_{l_1,(m,n)}}{(1-\gamma_{l_1,(m,n)}) b_{(m,n)} - X_{l_2,(m,n)}} = q, \quad \text{for each } (m,n). \tag{5.7}$$

WFQ-weight $\gamma_{l_1,(m,n)}$ can be solved from equation (5.7) as a function of traffic of both classes.

$$\gamma_{l_1,(m,n)}(X_{l_1,(m,n)}, X_{l_2,(m,n)}) = \frac{b_{(m,n)} + q X_{l_1,(m,n)} - X_{l_2,(m,n)}}{b_{(m,n)}(q+1)}, \quad \text{for each } (m,n). \tag{5.8}$$

The optimization function of the presented flow allocation problem is almost similar to equation (5.5). The difference is that the WFQ-weight is not a free parameter in the optimization where the ratio of link delays are fixed. The optimization problem is as follows:

$$
\begin{aligned}
&\text{Minimize} \\
&\sum_{l} w_l E[D_l] \\
&= \frac{w_{l_1}}{\Lambda_{l_1}} \sum_{(m,n)} \frac{X_{l_1,(m,n)}}{\gamma_{l_1,(m,n)} b_{(m,n)} - X_{l_1,(m,n)}} \\
&+ \frac{w_{l_2}}{\Lambda_{l_2}} \sum_{(m,n)} \frac{X_{l_2,(m,n)}}{(1-\gamma_{l_1,(m,n)}) b_{(m,n)} - X_{l_2,(m,n)}}, \\
&\text{subject to the constraints} \\
&\gamma_{l_1,(m,n)} = \frac{b_{(m,n)} + q X_{l_1,(m,n)} - X_{l_2,(m,n)}}{b_{(m,n)}(q+1)}, \quad \text{for each } (m,n), \\
&X_{l,(m,n)} < \gamma_{l,(m,n)} b_{(m,n)}, \quad \text{for each } (m,n), \\
&A x^T_{l,(i,j)} = R^T_{l,(i,j)}, \quad \text{for each } l,(i,j), \\
&0 < \gamma_{l,(m,n)} < 1, \quad \text{for each } l,(m,n), \\
&\sum_{l} \gamma_{l,(i,j)} = 1, \quad \text{for each } (i,j).
\end{aligned}
\tag{5.9}
$$

If we want to differentiate the classes by fixing the link delays only, the cost weights of the optimization function are equal to one (referred to as "fixed link delays"). We can also optimize the sum of the weighted mean delays (referred to as "fixed link delays and weights"). The problem is now how to determine the cost weights in relation to the ratio of link delays.

Chapter 6

Numerical results

6.1 Introduction

We have implemented three different load balancing algorithms, minimum-delay using non-linear optimization (referred to as "minimum-delay"), minimum-delay using two-level procedure (linear and non-linear optimization, referred to as "LP-NLP") and minimum-delay using heuristic approach (referred to as "heuristics"). We have tested the heuristic algorithm using different level of granularity. All three methods have been compared.

The routing methods that try to differentiate the mean delay between traffic classes, are tested also. In order to make optimizations simpler, we study only the case of two classes, gold and silver. The traffic matrices of both classes are equal, so that the comparison between the traffic classes is easier. The traffic demands in traffic matrices are a half of the original demands.

The formulations of the optimization problems were written using General Algebraic Modelling System (GAMS), which is a high-level modelling language for mathematical programming problems. It is specially useful for complex and large scale linear and non-linear optimization. GAMS-program calls a solver, which can be specified in the program. We have used solver module Minos 5 in our optimizations.

The algorithms are tested in a well-known test-network, which consists of 10 nodes, 58 links and 72 ingress-egress pairs. The test-network is generated by Curtis Villamizar for testing MPLS-OMP [Vil99b]. The topology of the network is presented in Figure 6.1. The link capacities and the traffic demands of the ingress-egress pairs are presented in Appendix A.

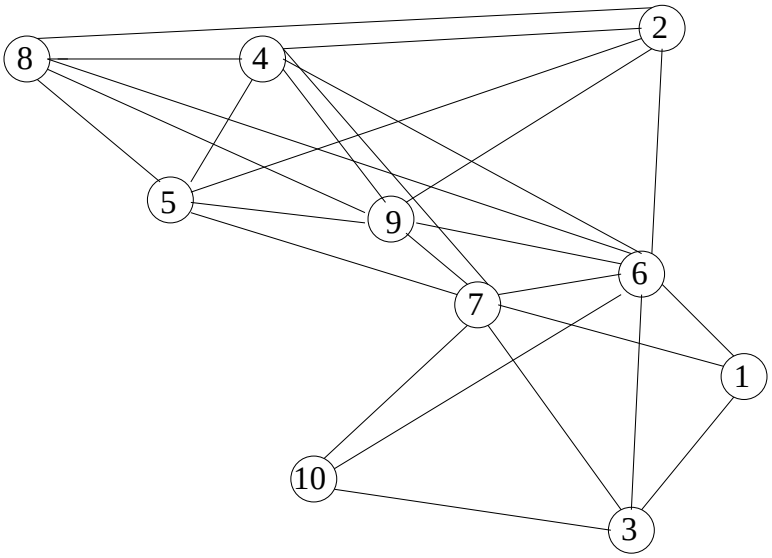

Figure 6.1: The test-network

6.2 Load-balanced routing

6.2.1 The performance of heuristics using different levels of granularity

The heuristics described in section 3.4 is implemented and tested using Mathematica. We have used granularity levels 1, 2, 4, 8, 16 and 32, which correspond to the lengths of the prefixes 0, 1, 2, 3, 4 and 5.

The results in Figure 6.2 show that the level of granularity has a great impact on the mean delay. On the other hand, the use of a finer granularity than 8 does not decrease mean delay significantly. The policy to route first an ingress-egress pair that has the greatest traffic intensity seems to result in the smallest mean delay. The explanation to this is that the ingress-egress pairs with the greatest traffic intensity are the hardest problems to route and there is more free capacity and therefore more routing alternatives in the network in the early phase of the algorithm.

Next we have increased the traffic load of each ingress-egress pair. The mean delay as a function of the percentage of the traffic load that uses all available capacity using different levels of granularity is presented in Figure 6.3. The greater the traffic load, the greater the impact of the level of granularity on the mean delay.

We notice that the routing problem is infeasible when the level of granularity is small but the traffic load is great. For example, when granularity level 1 is used we can route approximately 26% less traffic than by using granularity level 8. The maximum loads that can be routed using a particular level of granularity are presented in Table 6.1

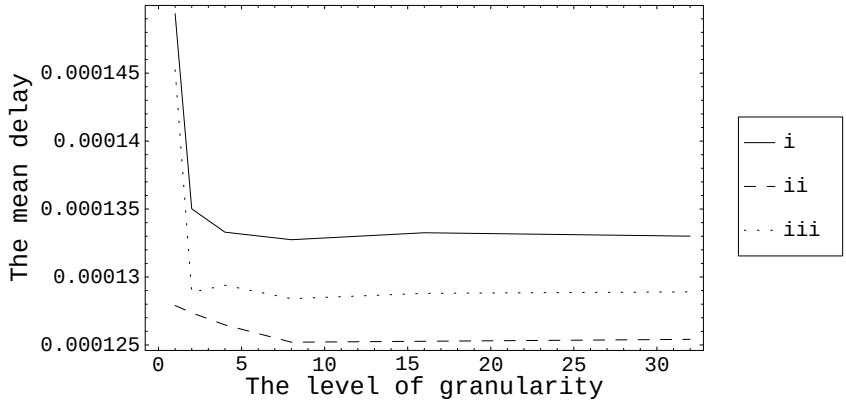

Figure 6.2: The mean delay as a function of granularity when streams are routed in (i) increasing order in terms of the traffic load, (ii) decreasing order in terms of the traffic load, and (iii) decreasing order in terms of the mean delay.

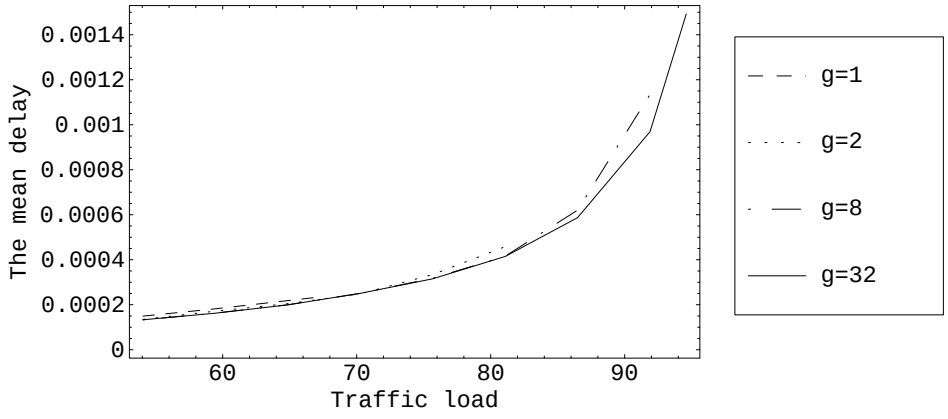

Figure 6.3: The mean delay as a function of traffic load using the granularity level 1, 2, 8 and 32

Table 6.1: The maximum loads that can be routed using a particular level of granularity

The level of granularity	The maximum load (%)
1	67.6
2	81.1
8	91.9
32	94.6

6.2.2 Comparison of load-balanced routing methods

The mean delay as a function of the traffic load of the three routing methods is presented in Figure 6.4. Figure 6.5 presents the relative deviation of the mean delay from the optimal as a function of the traffic load. With the heuristic approach, we use the granularity level 32, except in the cases of heavy load (the traffic load is over 95%) when the used granularity level is 128. We can see that the mean delays of different methods do not differ significantly. Only when the traffic load is near to the total capacity of network, is the performance of the minimum-delay routing notable. The mean delay using LP-NLP-routing and heuristics are quite similar (see Figure 6.5), until the load is heavy, when LP-NLP-optimization provides a better result.

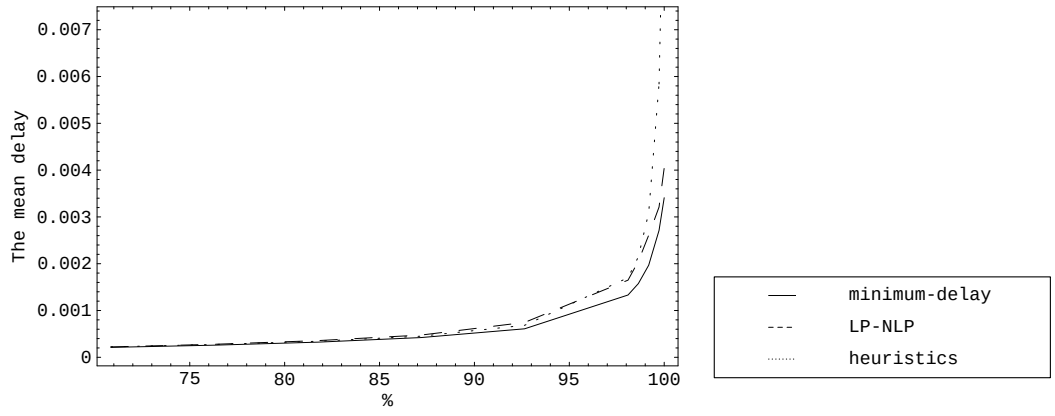

Figure 6.4: The mean delay as a function of the traffic load

The difference between the minimum-delay routing and LP-NLP routing is that the set of paths is reduced in the latter approach. The number of selected paths has an effect on the mean delay obtained in the second phase. If there are few paths, the flow allocation is far from optimal. On the other hand, the computation time of the second phase is proportional to the number of paths. In Table 6.2 the number of ingress-egress pairs that use a particular number of paths is presented in the case where the load of traffic is heavy (98% of the traffic in the situation, where the load exceeds the bandwidth). Both in the minimum-delay routing and the LP-NLP-routing, the maximum number of paths is three. However, the

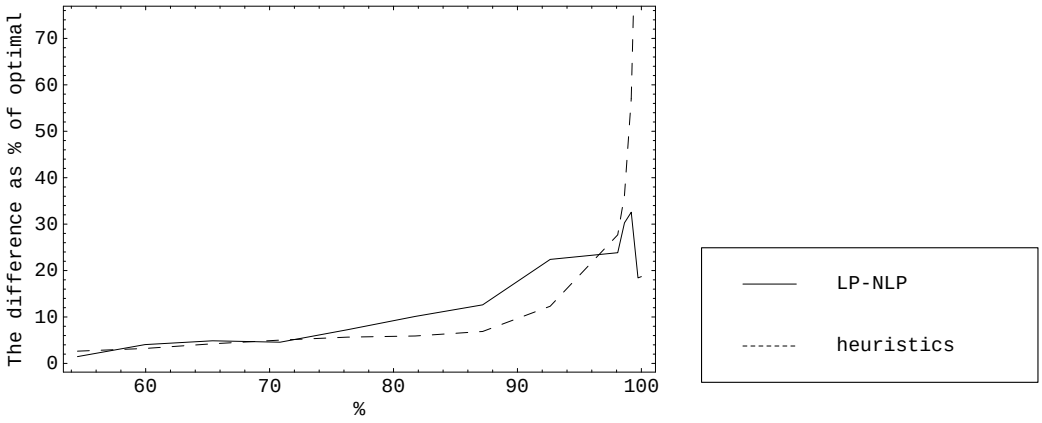

Figure 6.5: The relative deviation of the mean delay from the optimal as a function of the traffic load

number of ingress-egress pairs that use three paths is greater in the minimum-delay routing. This makes the routing more flexible and the mean delay smaller.

Table 6.2: The number of ingress-pairs that use 1, 2 or 3 paths

minimum-delay	52	13	7
LP-NLP routing	57	14	1

The computation times of the LP-, NLP- and minimum-delay optimizations are compared in Table 6.3. We exclude the heuristic approach from the comparison because the implementation softwares are different. Neither is the time to solve paths from the flow allocation solution taken into account in the case of the LP-NLP-routing. The table shows that the computation time of the LP-optimization is more than ten times longer than that of the NLP-optimization. The reason is that in the NLP-optimization the number of free variables is reduced and is the same as the number of the paths (only 88 variables in the previous example). So it is reasonable to use the LP-calculation in the short time scales in order to respond to small changes in the traffic demand matrix, as was suggested in [Ott01]. However, the performance in short time scales can be poorer than Figure 6.5 shows because the paths are calculated using some former traffic matrix.

Table 6.3: The computation times of different optimizations in seconds

Optimization	Generation time	Execution Time
LP-optimization	0.27	0.27
NLP-optimization	0.02	0.02
minimum-delay	3.3	3.3

Table 6.3 shows also that the computation time of the LP-NLP-optimization is ten times shorter than the computation time of the minimum-delay optimization. So

the performance of the LP-NLP-approach is very good if the total traffic demand does not exceed 70% of the maximum load.

6.2.3 Bottleneck problem in LP-NLP-optimization

In some cases the performance of the LP-NLP-optimization is very poor. Consider a network where some link is a bottleneck. The LP-optimization tries to balance load by minimizing the maximum link load. Parameter Z of equation (3.17) is determined by the bottleneck. After Z is fixed, the latter term of the optimization function of (3.17) is minimized by minimizing the hop-count. This may lead to a small number of paths and bad performance when NLP-optimization is done.

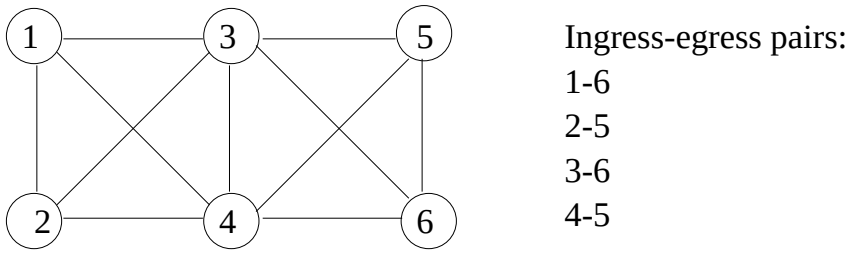

Figure 6.6: The network example

An example network of the bottleneck problem is presented in Figure 6.6. The capacities of all links are 100 and the traffic demands of all ingress-egress pairs are 90. One flow allocation that minimizes the optimization function of (3.17) is presented in Table 6.4. The mean delay calculated using (3.27) is 0.15. A simple alternative flow allocation can be obtained using heuristic approach, where the level of granularity is 2. In this allocation, ingress-egress pair $1-6$ is routed so that traffic is divided equally into two paths, $1-4-6$ and $1-2-4-6$. Now the mean delay would be 0.13 that is almost 12% smaller delay than the delay achieved using LP-NLP-routing.

Table 6.4: The flow allocation of the LP-optimization

Ingress-egress pair	path
1-6	1-4-6
2-5	2-3-5
3-6	3-6
4-5	4-5

The enhancement of the LP-NLP-optimization presented in section 3.3.4 selects additional paths to the solution in advance. This approach may improve the performance of the LP-NLP-algorithm in the bottleneck cases.

6.3 Methods to achieve service differentiation

6.3.1 Routing methods relying on routing only

The first algorithm in section 5.2 makes an attempt to differentiate the classes by optimizing the sum of weighted mean delays. As a result, the ratio of the mean delay of the silver class to the mean delay of the gold class as a function of the cost weight of the gold class is presented Figure 6.7. We can see that the ratio of mean delays increases when the cost weight of the gold class increases and the cost weight of the gold class should be great in order to achieve differentiation.

Figure 6.7: The ratio of mean delays as a function of the cost weight of the gold class

In the case where the traffic demand matrices of each class are equal, the flow allocation that differentiates the mean delay differs from the flow allocation of the optimal routing. So the mean delay increases when the cost weight of the gold class increases. The growth of the mean delay as a function of the cost weight of the gold class is presented in Figure 6.8.

Next we have implemented the algorithm that fixes the ratio of mean delays. The implementation of the algorithm is straightforward because the algorithm does not require cost weights.

The heuristic approach has also been implemented. The heuristics routes first the gold class and multiplies the allocated traffic by some factor in order to make the silver class to avoid the links used by gold class. The ratio of mean delay as a function of multiplier $(1 + \Delta)$ is presented in Figure 6.9. The maximum ratio that can be obtained using this approach is approximately 2, which may be too small for the required differentiation.

We take the mean delay of the total network as a function of the ratio of mean delay as a performance indicator. The increase in mean delay describes the cost

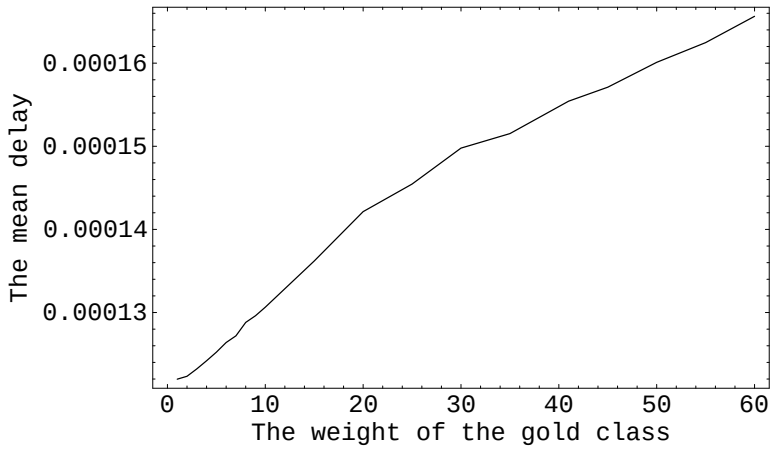

Figure 6.8: The mean delay of the network as a function of the cost weight of the gold class

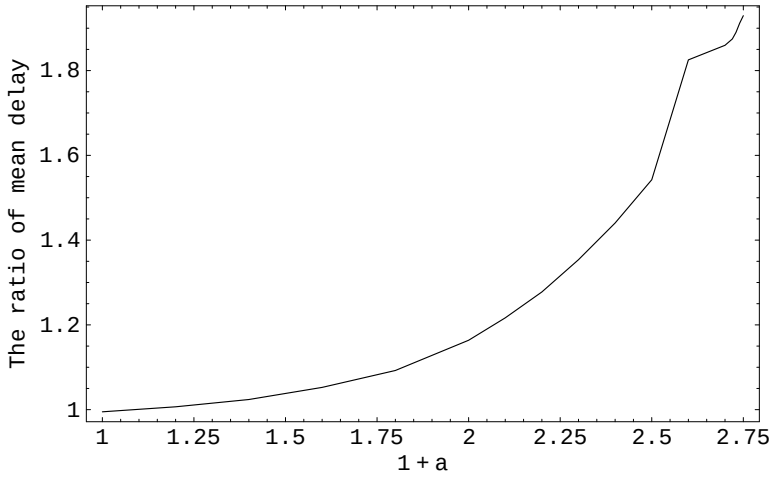

Figure 6.9: The ratio of mean delays as a function of $(1 + \Delta)$

of achieving a certain level of differentiation. All three methods are compared in Figure 6.10.

The figure shows that the first and second optimizations generate the same result. This can be understood by considering the situation where the ratio of mean delay is fixed to parameter q. The optimization function of (5.2) can be reduced to

$$w_{l_1} E[D_{l_1}] + w_{l_2} E[D_{l_2}] = (w_{l_1} + q w_{l_2}) E[D_{l_1}], \tag{6.1}$$

the minimum value of which can be determined explicitly for each w_{l_1} and w_{l_2}. However, the benefit of optimization function (5.3) is that the ratio of mean delays is known a priori, while the cost weights of optimization function (5.2) must be determined.

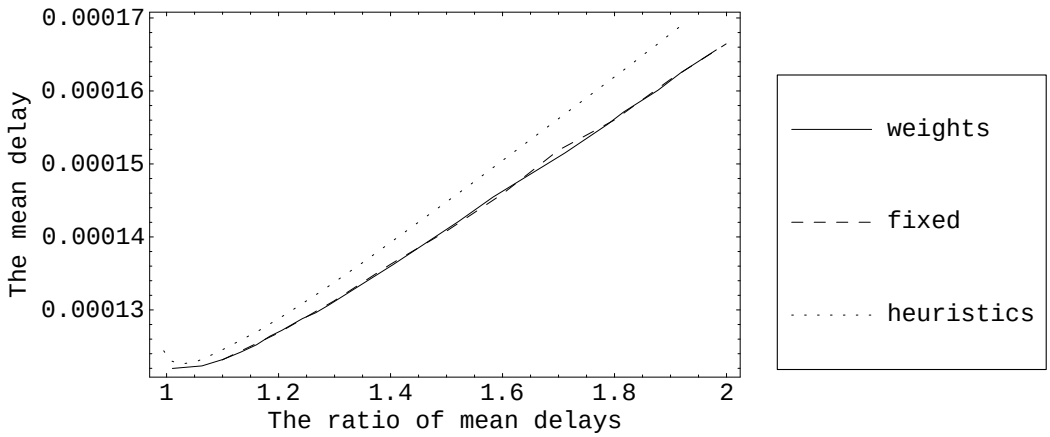

Figure 6.10: The mean delay of the network of as a function of the ratio of mean delays

6.3.2 Routing methods making use of WFQ-scheduling

Optimization using cost weights

We have implemented the optimization method that minimizes the weighted sum of mean delays straightforwardly and the optimization methods that divide the problem into two steps. Both the straightforward algorithm and the two-step approaches were introduced in section 5.3.1.

The first two-step algorithm optimizes the flow allocation and then optimizes the WFQ-weights. The flow allocation without WFQ-weights can be done optimally, when each class achieves the same amount of bandwidth ("two-step, version 1"). On the other hand, the flow allocation can be done by taking the classes into account and using the optimizing function presented in section 5.2.1 ("two-step, version 2").

A near optimal routing can also be achieved using an iterative approach (referred to as "two-step, iterative"). The flow allocation and the WFQ-weights are optimized alternatingly. The change of the value of the optimization function as a function of the number of iterations is presented in Figure 6.11. In the following optimizations we use the minimum-delay routing as the starting point ("two-step, version 1"). The number of iterations is ten, which means that ten flow allocations and ten WFQ-weight determinations are done in the optimization.

We have also implemented an algorithm that first calculates paths using the LP-optimization and after that allocates traffic and determines the WFQ-weights using the NLP-optimization ("QoS-LP-NLP").

The ratios of the mean delay of the silver class to the mean delay of the gold

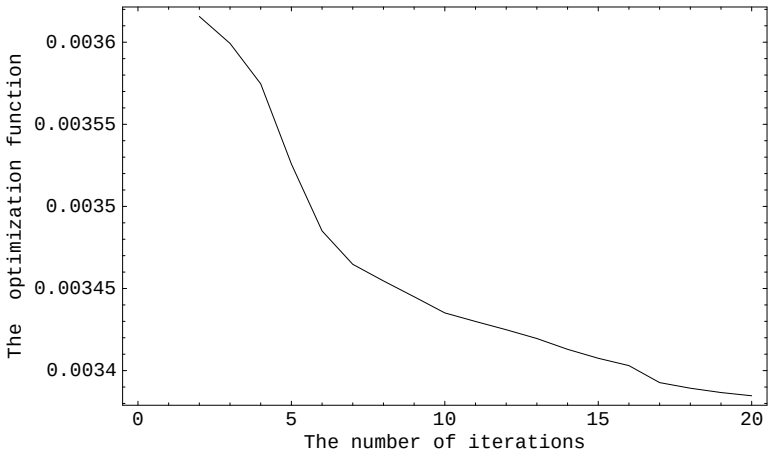

Figure 6.11: The objective function as a function of the number of iterations

class of all five algorithms are presented in Figure 6.12. In Figure 6.13 we present the mean delay as a function of the weight of the gold class. The numbering of curves in the previous and following figures is explained in Table 6.5. All methods except the straightforward routing behave equally. The irregularities in the curve of the straightforward routing are perhaps a consequence of numerical errors in the optimization procedure. As the result, the cost weight to achieve a certain ratio of mean delay is more than ten times smaller in the optimization that makes use of WFQ-weights than in the optimization without WFQ-weights.

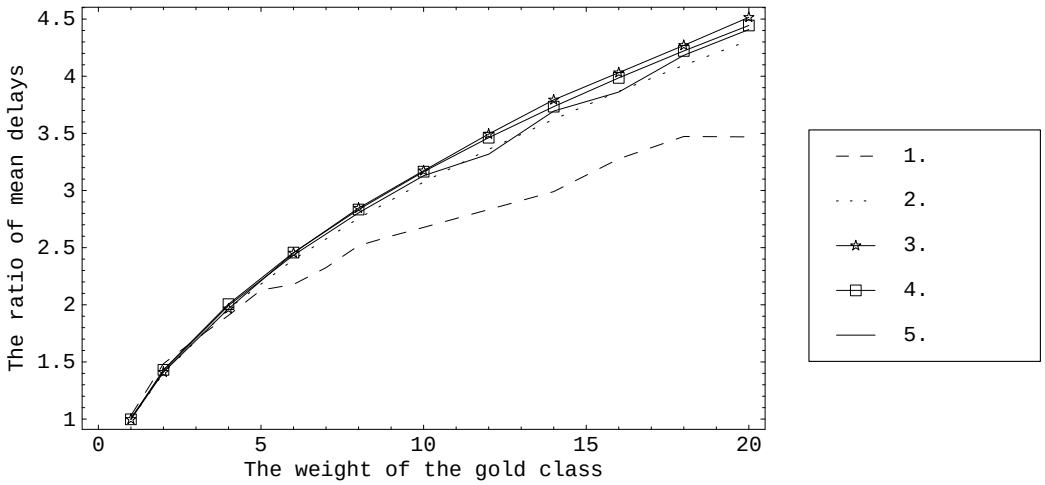

Figure 6.12: The ratio of mean delay as the function of the cost weight of gold class

Table 6.5: The numbering of the algorithms used

	Algorithm
1.	straightforward
2.	two-step, version 1
3.	two-step, version 2
4.	two-step, 10 iterations
5.	QoS-LP-NLP
6.	fixed link delays
7.	fixed link delays and weights

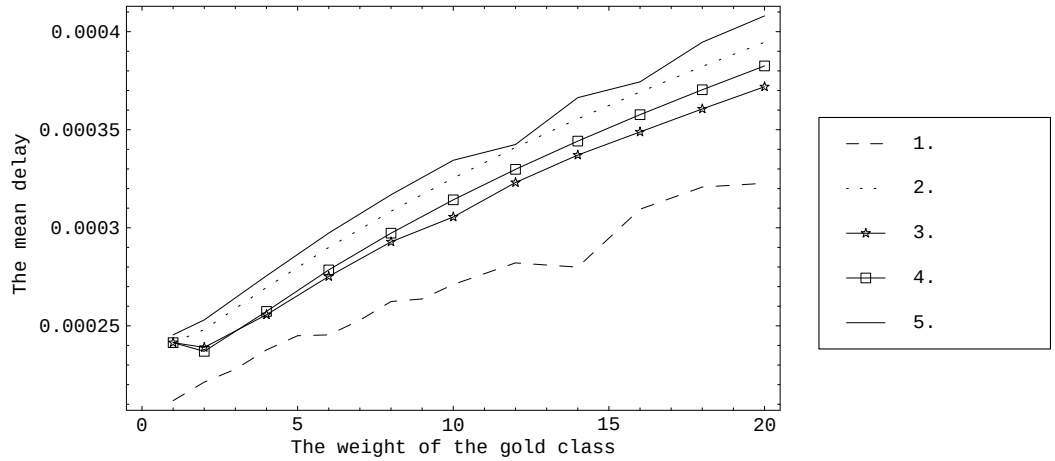

Figure 6.13: The mean delay of the network as a function of the cost weight of the gold class

Finally, we study the effect of changes in offered traffic on the ratio of mean delays. We use "two-phase, version 2" as the routing algorithm, with the cost weight of the gold class fixed to 10. We increase the offered traffic of both the gold and the silver classes equally until the offered traffic exceeds the capacity of the network. As it can be seen from Figure 6.14, the ratio of mean delays varies only slightly, when the traffic load changes.

Optimization by fixing the ratio of link delays

In order to simplify optimization and to provide differentiation also at the ingress-egress pair level, we have implemented optimizations that fix the ratio of link delays to some parameter q ("fixed link delays"). It is not guaranteed that the ratio of mean delays of different classes is the same as the ratio of link delays. The optimization problem is presented in section 5.3.2

We combine also cost weight w_{l_1} to the optimization function with the fixed ratio of link delays ("fixed link delays and weights"). In this approach the problem

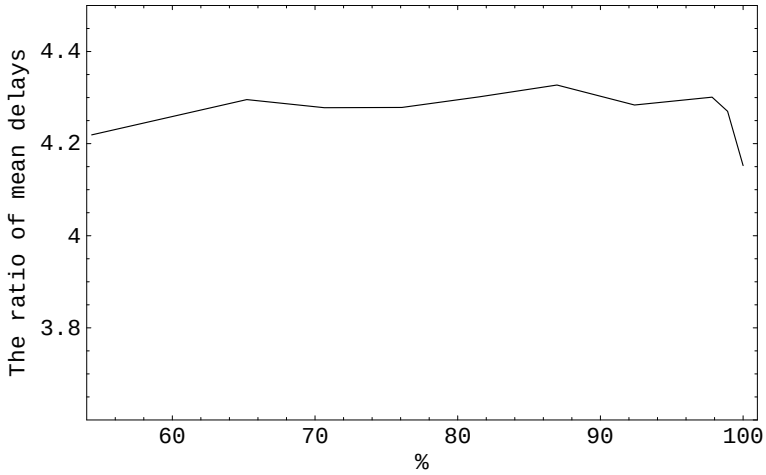

Figure 6.14: The ratio of mean delays as a function of the percentage of maximum traffic load

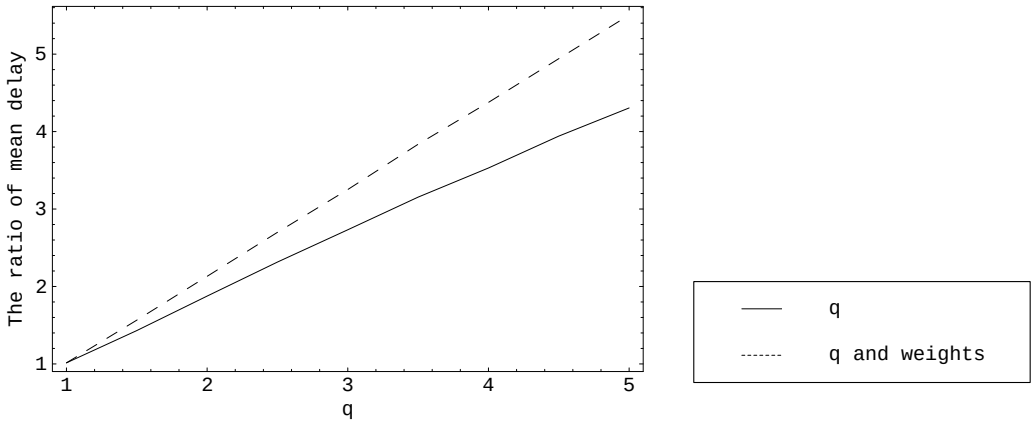

Figure 6.15: The ratio of mean delays as a function of q

is how to determine both the parameter q and the cost weight w_{l_1}. We have implemented only one case, where the cost weight is three times greater than link delay ratio q.

In Figure 6.15 we present the relation between the ratio of link delays and the ratio of mean delays of different classes. The ratio of mean delays seems to be smaller than the ratio of link delays. The explanation is that the routing algorithm tries to balance traffic load by routing classes that achieve more bandwidth through long routes. The routing that uses cost weights also seems to provide a greater ratio of mean delays than the routing that uses only parameter q. The reason is that the routing with cost weights utilizes also routing when trying to differentiate the classes.

As expected, the use of a fixed link delay provides differentiation also at the ingress-egress pair level, when the traffic matrices are equal. To make a comparison, in Table 6.6 we present the percentage of ingress-egress pairs of the silver

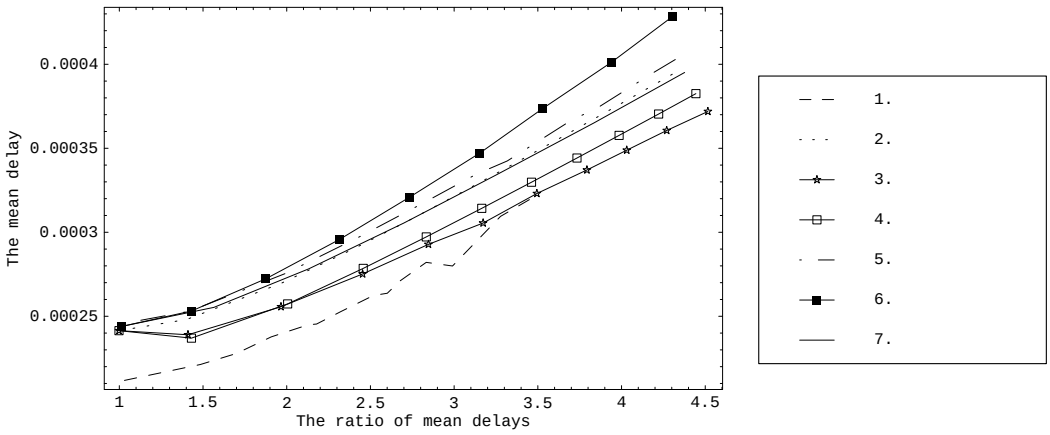

Figure 6.16: The mean delay of the network of as a function of the ratio of mean delays

class that have smaller mean delay than the corresponding pair of gold class. In the case of fixed link delays, the percentage is always zero. When the cost weight of the gold class is greater than eight, the percentage is also zero in the case of two-step algorithms. The percentages are greater in "two-step algorithm, version 2" than in "two-step algorithm, version 1". The explanation is related to the fact that in the second version of the algorithm the classes are differentiated already in the first phase by routing the silver class along longer routes than the gold class.

Table 6.6: The percentage of ingress-egress pairs of silver class that have smaller mean delay than the corresponding pairs of gold class

w_{l_1}	two-step, version 1	two-step, version 2	fixed link delays
2	1.39	8.33	0
4	0	2.78	0
6	0	1.39	0
8	0	0	0
10	0	0	0
12	0	0	0

Conclusions on the methods

Finally, we compare all the methods. The performance metric is the same as in section 6.3.1, the mean delay of total network as a function of the ratio of mean delays. The results are presented in Figure 6.16.

The straightforward optimization seems to have the smallest mean delay. The difference to other algorithms is significant when the ratio of mean delays is small.

Table 6.7: The computation times of optimization algorithms in seconds

Algorithm	The computation time
straightforward	25.22
two-step, version 1	24.98
two-step, version 2	24.90
two-step, 10 iterations	255.0
QoS-LP-NLP	1.34
fixed link delays	25.56
fixed link delays and weights	25.33

When the ratio is greater, the performance of the two-step algorithm that utilizes both routing and WFQ-weights ("two-step, version 2") is near to optimal.

The performance of the two-step algorithm that allocates the traffic using minimum-delay routing ("two-step, version 1") seems to be worse than the performance of the "two-step, version 2". The explanation is that the latter version makes use of both routing and WFQ-scheduling and is therefore more flexible. The performance of "two-step, 10 iterations" is in the middle of "two-step, version 1" and "two-step, version 2". However, all these three two-step algorithms that allocate first traffic have a better performance than the "QoS-LP-NLP"-algorithm.

The performance of the algorithm that fixes the ratio of link delays is not very good. The performance improves significantly if the cost weights are determined in some manner.

As a conclusion, the differentiation methods that make use of both routing and WFQ-scheduling ("straightforward", "two-step, version 2", "two-step, iterative", "fixed link delays and weights") perform better than the differentiation methods that make use of only WFQ-scheduling ("two-step, version 1", "QoS-LP-NLP" and "fixed link delays").

The computation times of the optimization problems are presented in Table 6.7. All times are quite similar, except for the computation times of the iterative approach and the "QoS-LP-NLP"-optimization. The iterative algorithm iterates the two-step optimization 10 times, so the computation time is also ten times greater than the computation time without iterations. The computation time of the "QoS-LP-NLP"-optimization is very short, but the routing obtained by using this method is quite far from optimal.

Chapter 7

Conclusion

7.1 Summary

MPLS is a flexible new technique that is designed to provide technical capabilities for the future Internet with various users and various services. Its key benefit is the support for Traffic Engineering that enables the use of explicit routes, for example. Explicit routing gives the capability to route traffic along paths that differ from those selected by IP routing. Traffic can also be split to several paths and load can thus be balanced.

Several load balancing algorithms have been introduced in the literature. We concentrated on three methods that try to minimize the mean delay of the network. The first algorithm is minimum-delay routing presented by Gallager. The second algorithm calculates first paths using the LP-optimization and after that allocates traffic using the NLP-optimization. The third algorithm is a heuristic approach that divides traffic into parts and routes them consecutively using Dijkstra's algorithm. As a result, we notice that the computation time of the two-step algorithm is ten times smaller than the computation time of the optimal routing. However, the mean delays of algorithms differ significantly only if the load is near to the maximum.

We have also studied the effect of the level of granularity by the heuristic algorithm. We notice that the mean delay is great when the level of granularity is under eight (the traffic is divided into eight parts). After that, when the level of granularity is increased, the mean delay does not decrease significantly. However, if we increase the traffic load, the routings that use low level of granularity become infeasible.

We have used load balancing algorithms as a starting point when developing routing methods that try to differentiate traffic classes in terms of the mean delay. First we have developed three algorithms that differentiate classes by

routing only. The first one minimizes the sum of weighted mean delays. The second one fixes the ratio of mean delays. The third one is a heuristic approach that routes the higher priority class first and multiplies the allocated traffic by some factor. After that the lower priority class is routed. We use the overall mean delay as a function of the ratio of mean delays as a performance indicator. The performances of the first and second algorithms are equal. The disadvantage of the first algorithm is that the cost weights have to be known in advance. The performance of the heuristic approach is a little poorer than that of the other two algorithms.

WFQ-scheduling provides guaranteed bandwidth sharing. We have presented a model where the bandwidth of each link is shared among the traffic classes according to the WFQ-weights. The optimization problem is to minimize the mean delay. We have done this by minimizing the weighted sum of mean delays and using two-step approaches. The first approach allocates traffic and after that determines the WFQ-weights. The second approach calculates first paths using the linear optimization and after that determines the flow allocation and the WFQ-weights using non-linear optimization. The fourth approach fixes the ratio of link delays. We notice that the use of the algorithm that makes use of both routing and WFQ-scheduling gives the best result. However, the computation time of the two-step algorithm that uses both linear and nonlinear optimization is twenty times smaller than the computation time of the other algorithms.

7.2 Further work

Both MPLS and Differentiated Services are under an aggressive development process today. Many proposals are made and will be made for practical implementation of these fields. Keeping track of new techniques is important.

As to the load balancing methods, these should be tested in various networks. The performance of the routing algorithms depends on the topology and size of the network. The granularity of heuristics should be studied more, how it could be combined to some optimization methods, for example. The cost of splitting traffic into arbitrary fractions should be compared to the cost of the increase in the mean delay when traffic allocation is restricted by some level of granularity. The complexity of calculation of the heuristic approach should be studied and the algorithm should be implemented on the same platform as the other methods in order to make the comparison of the computation times easier.

Also routing methods that try to differentiate mean delays should be tested in various networks. For example, the routing methods can behave differently if there are fewer links. The test-network that we used in our optimizations, includes 52 links, while the maximum number of links is 90 (there exists a link from every node to every other node). In our optimization there were only two classes, and

their traffic matrices were assumed equal. It would be interesting to increase the number of traffic classes and study how the results differ if the traffic demand matrices are unequal.

The bandwidth guaranteed by WFQ-scheduling to each class is the theoretical minimum. WFQ as a work-conserving discipline serves packets of a class regardless of WFQ-weights if the queues of the other classes are empty. The actual bandwidth depends on the nature of traffic. As a result, the ratio of mean delays may differ from the result obtained by the optimizations. It would be interesting to know whether the actual ratio of mean delays is greater or smaller. A simulation study of WFQ-scheduling may provide some answers. Another question is how the actual bandwidth could be included in the optimization problems.

Bibliography

[And01] E. Anderson , T. Anderson, S. Gribble, A. Karlin and S. Savage, A Quantitative Evaluation of Traffic-Aware Routing Strategies, Student poster session, ACM SIGCOMM 2001, to be included in ACM Computer Communications Review.

[Andr99] I. Andripoulos and G. Pavlou, Supporting Differentiated Services in MPLS Networks, Proceedings of the 7th IEEE/IFIP Workshop on Quality of Service (IWQoS '99), London, UK, pp. 207-215, ISBN 0-7803-5671-3, IEEE, 1999.

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

[Ben96] J. Bennett and H. Zhang, Hierarchical Packet Fair Queueing Algorithms, IEEE/ACM Transactions on Networking, 5(5):675-689, Oct 1997. Also in Proceedings of SIGCOMM'96, August, 1996.

[Bert92] D. Bertsekas and R. Gallager, Data Networks, Prentice Hall, Englewood Cliffs, N.J., 2nd edition, 1992.

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

[Cas94] J. Castro and N. Nabona, Computational tests of a nonlinear multicommodity network flow code with lienar side contraints through primal partitioning, DR 94/05, Statistics ans Operations Research Dept., Universitat Politéctica de Catalunya, Barcelona.

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

[Fau02] F. Faucheur, M. Tatham, T. Telkamp, J. Boyle, W. Lai, P. Hicks, A. Chiu, W. Townsend and D. Skalecki, Requirements for support of Diff-serv-aware MPLS Traffic Engineering, Internet draft <draft-ietf-tewg-diff-te-reqts-04.txt>, October 2002.

[Fer98] P. Ferguson and G. Huston, Quality of Service; Delivering QoS on the Internet and in Corparate Networks, John Wiley & Sons, Inc., 1998.

[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.

[Nor02] S. Norden and J. S. Turner, New Inter-domain QoS Routing Algorithms, Technical Report, Dept. of computer Science, WUCS-02-03, March 2002.

[Gué99] R. Guérin and V. Peris, Quality-of-Service in packet networks: basic mechanisms and directions, Computer Networks, Vol. 31, No. 3, pages 169-179, February 1999.

[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.

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

[RFC2205] R. Braden, L. Zhang, S. Berson, S. Herzog and S. Jamin, Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification, IETF RFC 2205, September 1997.

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

[RFC2430] T. Li and Y. Rekhter, A Provider Architecture for Differentiated Services and traffic Engineering (PASTE), IETF RFC 2430, October 1998.

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

[RFC2597] J. Heinänen, F. Baker, W. Weiss and J. Wroclawski, Assured Forwarding PHB Group, IETF RFC 2597, June 1999.

[RFC2598] V. Jacobson, K. Nichols, K. Poduri, An Expedited Forwarding PHB, IETF RFC 2598, June 1999.

[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.

[RFC3213] J. Ash, M. Girish, E. Gray, B. Jamoussi and G. Wright, Applicability Statement for CR-LDP, IETF RFC 3213, January 2002.

[RFC3214] J. Ash et al., LSP Modification using CR-LDP, IETF RFC 3214, January 2002.

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

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

[Spr00] S. Spraggs, Traffic Engineering, Carrier-scale IP networks, BT exact communications technology series 1, Vol 18, No 3, pages 235-260, July 2000.

[Seg77] A. Segall, The Modelling of Adaptive Routing in Data Communication Networks, IEEE Transactions on Communication, 27:85-95, January 1977.

[Sem] C. Semeria, Supporting Differentiated Service Classes: Queue Scheduling Disciplines, White paper, Juniper Networks.

[Srid00] A. Sridharan, S. Bhattacharyya, R. Guérin, J. Jetcheva and N. Taft, On The Impact of Aggregation on The Performance of Traffic Aware Routing, Technical report, University of Pennsylvania, July 2000.

[Vil99a] C. Villamizar, OSPF Optimized Multipath (OSPF-OMP), Internet draft<draft-villamizar-ietf-ospf-omp-02>, February 1999.

[Vil99b] C. Villamizar, MPLS Optimized Multipath (MPLS-OMP), Internet draft <draft-villamizar-mpls-omp-01>, 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.

[Vut00] S. Vutukury and J.J. Garcia-Luna-Aceves, A Traffic Engineering Approach based on Minimum-Delay Routing, Proc. IEEE IC3N 2000, USA, October 16–19, 2000.

[Wan01] Z. Wang, Internet QoS, Architecture and Mechanisms for Quality of Service, Morgan-Kaufman Publishers, USA, 2001.

[Wid98] I. Widjaja, A. Elwalid, MATE: MPLS Adaptive Traffic Engineering, Internet draft <draft-widjaja-mpls-mate-00.txt>, August 1998.

[Xia99] X. Xiao and L. M. Ni, Internet QoS: A Big Picture, IEEE Network, March/April 1999.

[Zau99] J.J. Garcia-Luna-Aceves, S. Vutukury, W. Zaumen, A Practical Approach to Minimizing Delays in Internet Routing, Proc. IEEE ICC '99, Canada, June 6–10, 1999.

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

[Zha95] H. Zhang, Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks, Proceedings of the IEEE, Vol. 83, No 10, October 1995.

Appendix A: The test-network

We have used the test-network from web-page http://brookfield.ans.net/omp/random-test-cases.html in our optimizations. The test-network is generated by Curtis Villamizar for testing MPLS-OMP [Vil99b].

Table 7.1: The links

	m	n	$b_{(m,n)}$		m	n	$b_{(m,n)}$
1.	1	3	16702	27.	6	4	22270
2.	1	6	16702	28.	6	7	22270
3.	1	7	16702	29.	6	8	20305
4.	2	4	16702	30.	6	9	17685
5.	2	5	16702	31.	6	10	17685
6.	2	6	17030	32.	7	1	16702
7.	2	8	16702	33.	7	3	21615
8.	2	9	16702	34.	7	4	24890
9.	3	1	16702	35.	7	5	20305
10.	3	6	16702	36.	7	6	26855
11.	3	7	16702	37.	7	9	16702
12.	3	10	16702	38.	7	10	16702
13.	4	2	16702	39.	8	2	16702
14.	4	5	16702	40.	8	4	16702
15.	4	6	18995	41.	8	5	16702
16.	4	7	20960	42.	8	6	17685
17.	4	8	16702	43.	8	9	16702
18.	4	9	16702	44.	9	2	16702
19.	5	2	16702	45.	9	4	16702
20.	5	4	16702	46.	9	5	16702
21.	5	7	24235	47.	9	6	16702
22.	5	8	16702	48.	9	7	17030
23.	5	9	16702	49.	9	8	16702
24.	6	1	16702	50.	10	3	16702
25.	6	2	33405	51.	10	6	18340
26.	6	3	28165	52.	10	7	18340

Table 7.2: The ingress-egress pairs

	i	j	$d_{(i,j)}$		i	j	$d_{(i,j)}$
1.	2	3	2333.63466	37.	6	7	7223.16158
2.	2	4	3968.16665	38.	6	8	4568.42644
3.	2	5	2601.30171	39.	6	9	4692.27312
4.	2	6	5052.06686	40.	6	10	3631.52335
5.	2	7	3983.25228	41.	7	2	4180.25640
6.	2	8	3192.23528	42.	7	3	3479.19051
7.	2	9	2953.08509	43.	7	4	5121.10119
8.	2	10	2743.23195	44.	7	5	3342.68726
9.	3	2	2324.40433	45.	7	6	6593.33824
10.	3	4	2972.14880	46.	7	8	4079.95930
11.	3	5	1990.66914	47.	7	9	3994.00236
12.	3	6	4322.77966	48.	7	10	3563.67980
13.	3	7	3039.28713	49.	8	2	3061.56332
14.	3	8	2445.78244	50.	8	3	2515.79314
15.	3	9	2592.62687	51.	8	4	4190.66487
16.	3	10	1852.68713	52.	8	5	2390.03354
17.	4	2	4132.45071	53.	8	6	5091.14910
18.	4	3	3153.58208	54.	8	7	4263.73290
19.	4	5	2968.23956	55.	8	9	2916.95979
20.	4	6	5740.23207	56.	8	10	2317.59273
21.	4	7	5670.65744	57.	9	2	2963.74805
22.	4	8	3628.48207	58.	9	3	2340.11237
23.	4	9	4009.99348	59.	9	4	3602.99269
24.	4	10	3273.57749	60.	9	5	2812.59983
25.	5	2	2752.43278	61.	9	6	4854.52280
26.	5	3	2085.55295	62.	9	7	3830.54090
27.	5	4	3138.30693	63.	9	8	3222.20899
28.	5	6	4125.33028	64.	9	10	2342.38275
29.	5	7	3742.27403	65.	10	2	2373.55848
30.	5	8	2628.68010	66.	10	3	1903.64130
31.	5	9	2823.66458	67.	10	4	3316.67524
32.	5	10	2165.21576	68.	10	5	1972.93541
33.	6	2	5345.47829	69.	10	6	3617.84978
34.	6	3	4155.11150	70.	10	7	3407.64460
35.	6	4	6738.86509	71.	10	8	2330.95132
36.	6	5	3834.99771	72.	10	9	2463.40334

Appendix B: Mathematica codes

B.1 Code to deduce paths from flow allocation

```
(*Algorithm to solve paths from flow allocation
    Author: Riikka Ssusitaival  <Riikka.Susitaival@hut.fi>
          Date:          27.8.2002*)

(*Algorithm to solve all paths from the ingress node to the egress node*)
AllRoutes[Ing_, Egr_, top_]:=
  Module[{Next,arc,l,a,newRoutes, l1, l2, l3, res, i,j,k},
    Next[arc_]:=
      Module[{l={}},
        For[i=0, i<Length[top],
          If[top[[i,1]]\[Equal]arc[[2]], l=Append[l, top[[i]]],i++];l];
    a={};
    For[i=0, i<Length[top],
      If[top[[i,1]]\[Equal]Ing, a=Append[a, {top[[i]]}],i++];
    newRoutes[l1_]:=Module[{l2,l3},
        l2=
          Table[If[l1[[i,Length[l1[[i]]],2]]\[NotEqual]Egr,
              Next[l1[[i,Length[l1[[i]]]]]],{i,1,Length[l1]}];
        l3=Flatten[
            Table[If[Length[l2[[i]]]\[NotEqual]0,
                Table[Partition[Flatten[{l1[[i]],l2[[i,k]]}],2],{k,1,
                    Length[l2[[i]]]}],{l1[[i]]}],{i,1,Length[l2]}],1];l3];
    res=FixedPoint[newRoutes,a];
    res]
(*Read the flow allocation from file*)
allo=Transpose[ReadList["U://minDelay18.dat",Table[Number,{72}]]];
(*Define the topology*)
kk=1.0;
rsTable=ReadList["D://test-case1.txt",Table[Number,{3}]];
dem=Table[rsTable[[i]],{i,1,72}];
dem=Table[{dem[[i,1]],dem[[i,2]], dem[[i,3]]*kk},{i,1,Length[dem]}];
dem1=dem;
cap=Table[rsTable[[i]],{i,73,Length[rsTable]}];
capTable=Table[Infinity,{i,1,10},{j,1,10}];
For[i=0,i<Length[cap],
    capTable=ReplacePart[capTable,cap[[i,3]],{cap[[i,1]],cap[[i,2]]}],i++];
topology={};
For[i=0,i<10,
  For[j=0,j<10,
    If[capTable[[i,j]]\[NotEqual] Infinity,topology=Append[topology,{i,j}]],
    j++],i++]
(*The ingress-egress pairs*)
For[i=0, i<Length[dem],dem=ReplacePart[dem, {dem[[i,1]],dem[[i,2]]},i],i++ ]
top=topology;
(*Reduce the unnecessery links from the topology*)
top=Table[
      If[allo[[i,j]]\[NotEqual]0,topology[[j]]],{i,1,Length[allo]},{j,1,
        Length[allo[[i]]]}];
intensity=
```

```
    Table[If[allo[[i,j]]\[NotEqual]0,allo[[i,j]]],{i,1,Length[allo]},{j,1,
        Length[allo[[i]]]}];
top=Table[DeleteCases[top[[i]], Null],{i,1, Length[top]}];
intensity=Table[DeleteCases[intensity[[i]], Null],{i,1, Length[intensity]}];
intensity=
    Table[If[allo[[i,j]]\[NotEqual]0,allo[[i,j]]],{i,1,Length[allo]},{j,1,
        Length[allo[[i]]]}];
top=Table[DeleteCases[top[[i]], Null],{i,1, Length[top]}];
intensity=Table[DeleteCases[intensity[[i]], Null],{i,1, Length[intensity]}];
(*Define the paths for each ingress-egress pair*)
allR=Table[AllRoutes[dem[[i,1]], dem[[i,2]], top[[i]]], {i,1, Length[dem]}];
alloTable=
    Table[Table[0,{k,1,Length[top[[i]]]}],{j,1,Length[allR[[i]]]}],{i,1,
        Length[dem]} ];
For[k=0, k<Length[allR],For[i=0, i<Length[allR[[k]]],
    For[j=0, j<Length[allR[[k,i]]],
      alloTable=
        ReplacePart[alloTable,1,
          Flatten[{k,Position[top[[k]],allR[[k,i,j]]],i}]],j++],i++ ],k++]
(*Solve path allocation x from equation Ax=b*)
lin=Table[
      PseudoInverse[alloTable[[i]]].intensity[[i]],{i,1,Length[alloTable]}];
(*As a result we get a path set*)
paths=Table[
      Table[If[lin[[k,i]]\[NotEqual]0,allR[[k,i]]],{i,1,
          Length[allR[[k]]]}],{k,1,Length[allR]}];
paths=DeleteCases[paths,Null];
```

B.2 Heuristic algorithm to balance load

```
(*Dijkstra's algorithm
    Author:       Jorma Virtamo  <Jorma.Virtamo@hut.fi>
          Modified by:  Vesa Timonen   <Vesa.Timonen@hut.fi>
                Date:     5.11.2001*)

dijkstra[d_,root_,dstn_:False,options___]:=
  Module[{core,prev,dist,cond,temp,new,m,md,x,y},
    core={root};
    prev={root};
    peri=Delete[Range[d//Length],root];
    dist={0};
    If[!IntegerQ[dstn],
      (* no destination *)cond:=(peri=!={}),
      (* destination dstn *)cond:=(dstn\[NotEqual]Last[core])];
    While[cond,
      {new,temp}=
        Transpose[(y=Infinity;
                MapIndexed[If[#1<y,y=#1;x=#2]&,#[[peri]]];{First[x],y})&
            /@ d[[core]]];
      md=Infinity;MapIndexed[If[#1<md,md=#1;m=#2]&,dist+temp];m=First[m];
      dist={dist,md}//Flatten;
      prev={prev,core[[m]] }//Flatten;
      core={core,peri[[m=new[[m]]]]}//Flatten;
      peri=Delete[peri,m]];
    If[IntegerQ[dstn], (* one destination dstn *)
      y=Position[core,dstn]//Flatten;x=Position[core,prev//Last]//First;
      While[First[y]\[NotEqual]1,y={x,y}//Flatten;
        x=Position[core,prev[[x]]//First]//First];
      {core,prev, dist}={core[[y]],prev[[y]],dist[[y]]};];
    {core,prev, dist}]

(*Heuristics to route traffic in descending order in terms of traffic \
intensity
    Author: Riikka Ssusitaival  <Riikka.Susitaival@hut.fi>
          Date:          27.8.2002*)
```

```
Heuristic1[gran_,band_,dem_]:=
    Module[{L,alloBand,dem1,index,routes,traffic,delayM,dijk,W},
      L=Length[band[[1]]];
      alloBand=Table[Table[0,{i,1,L}],{j,1,L}];
      dem1=dem;
      index=Position[dem1,Max[dem1[[All,3]]]][[1]];
      routes=Table[,{j,0}];
      dijkRoutes=Table[,{j,0}];
      traffic=dem1[[index[[1]],index[[2]]]]/gran;
      For[k=1,k\[LessEqual]Length[dem],
        routes=Append[routes,index[[1]]];
        For[kk=1,kk\[LessEqual]gran,

          delayM=Table[
              If[band[[i,j]]\[NotEqual]Infinity,
                If[band[[i,j]]\[GreaterEqual](traffic+alloBand[[i,j]]),
                  1/(band[[i,j]]-(traffic+alloBand[[i,j]])),Infinity],
                Infinity],{i,1,L},{j,1,L}];
          dijk=dijkstra[delayM,dem1[[index[[1]],1]],dem1[[index[[1]],2]]];

          If[!MemberQ[routes,{dijk[[1]],_}],
            routes=Append[routes,{dijk[[1]],traffic}];
            dijkRoutes=Append[dijkRoutes,dijk],
            routes=Table[
                If[Length[routes[[i]]]\[Equal]2 &&
                    routes[[i,1]]\[Equal]dijk[[1]],{routes[[i,1]],
                    routes[[i,2]]+traffic},routes[[i]]],{i,1,
                  Length[routes]}]];

          For[kkk=1,kkk<Length[dijk[[1]]],
            alloBand=
              Table[alloBand[[i,j]]+
                  If[i==dijk[[1,kkk-1]]&&j\[Equal]dijk[[1,kkk]],traffic,0],{i,
                  1,L},{j,1,L}],kkk++],
          kk++];
        dem1=
          Table[If[
              i==index[[1]],{dem1[[i,1]],dem1[[i,2]],0},{dem1[[i,1]],
                dem1[[i,2]],dem1[[i,3]]}],{i,1,Length[dem]}];
        index=Position[dem1,Max[dem1[[All,3]]]][[1]];
        traffic=dem1[[index[[1]],3]]/gran;
        k++];
      W={dijkRoutes,routes,
          Sum[(alloBand[[i,j]]/(band[[i,j]]-alloBand[[i,j]])),{i,1,L},{j,1,
              L}]};
      W];
```