Mälardalen University Licentiate Thesis
No.95

Internet Traffic Management

Henrik Abrahamsson

November 2008

School of Innovation, Design and Engineering
Mälardalen University
Västerås, Sweden

ISSN 1651-9256
ISBN 978-91-86135-08-9
Printed by Arkitektkopia, Västerås, Sweden
Distribution: Mälardalen University Press

Abstract

Internet traffic volumes continue to grow at a great rate. For network operators it is important to understand and manage the traffic behaviour in order to meet service-level agreements with their customers and to give end-users good communication performance.

This thesis considers three aspects of Internet traffic management: web traffic modelling, bandwidth allocation to TCP flows, and traffic engineering. The areas all have in common the need to understand and handle Internet traffic behaviour. For web traffic modelling the goal of the work itself is to understand traffic behaviour and to be able to generate realistic traffic in simulations and lab experiments. For traffic engineering and bandwidth allocation to TCP flows the purpose is to develop methods to steer and control the traffic.

The web is one of the most popular Internet applications. In order to understand how aggregated Internet traffic behaves and to be able to generate realistic traffic for simulations and lab experiments it is important to understand web traffic behaviour. This thesis presents a simple model of web client traffic. Starting from a packet trace of web traffic, we derive empirical probability distributions describing session lengths, time between user requests for web pages, and the amount of data that is transferred due to a single user request. Using these probability distributions we implement a web-client traffic generator and show that the generated traffic has the same characteristics as the original web traffic, including the traffic variability (self-similar properties).

TCP is the predominant Internet transport protocol. TCP is used by many popular applications including the web and it is used for transporting 80-90% of the Internet traffic. The second aspect of traffic management in this thesis is dynamic allocation of bandwidth to TCP flows. TCP provides a reliable flow of data between two hosts and adapts its rate to the available capacity. Network technologies such as Dynamic synchronous Transfer Mode (DTM) provides channels with dynamically adjustable capacity. The issue is to adap-

tively allocate bandwidth to TCP flows, when both TCP and the bandwidth allocation scheme can react to changes in the network load. We use simulation to investigate the behaviour of a bandwidth allocation scheme, its effect on TCP flows and on a network that can vary its capacity. The results show that the bandwidth allocation scheme usually works well for TCP flows but we also highlight a scenario where packet loss makes the feedback mechanisms interact in an unfavourable way.

The objective of traffic engineering is to avoid congestion in the network and to make good use of available resources by controlling and optimising the routing. The challenge for traffic engineering in IP networks is to cope with the dynamics of Internet traffic demands. This thesis propose l-balanced routings that route the traffic on the shortest paths possible but make sure that no link is utilised to more than a given level l, if possible. L-balanced routing gives efficient routing of traffic and controlled spare capacity to handle unpredictable changes in traffic. We present an l-balanced routing algorithm based on multi-commodity flow optimisation. We also present a heuristic search method for finding l-balanced weight settings for the legacy routing protocols OSPF and IS-IS. We show that the search and the resulting weight settings work well in real network scenarios.

Acknowledgements

First of all, I would like to thank my advisor Mats Björkman. Without his encouragement, guidance and support this thesis would never have been finished.

I am also very grateful to Adams Dunkels for his generosity and enthusiasm and for all the inspiring and helpful discussions we have had; first as fellow PhD students and then with Adam as my co-advisor.

I also want to thank Bengt Ahlgren for giving me the opportunity to work in the NETS group and to do research at SICS. I did my first research work and wrote the first paper under Bengts supervision and I learned a lot from that.

I also want to thank Anders Gunnar for many interesting discussions on all aspects of Internet traffic management and for great collaboration on many research papers and countless project deliverables.

Many thanks also to the other co-authors of the papers in this thesis: Ian Marsh, Olof Hagsand, Juan Alonso and Per Kreuger.

I also want to thank Björn Grönvall, Laura Feeney, and Javier Ubillos in the NETS group, and all other colleagues at SICS for their support and for creating a stimulating work environment.

Contents

I

Thesis

Chapter 1

Introduction

1.1 Internet – a network of networks

The Internet is a worldwide communication network that connects hundreds of millions of hosts and Internet users [1, 2]. It is a giant infrastructure of optical fibres, copper wires and wireless connections that via packet switches connect a wide variety of end-hosts: ranging from traditional web servers, PC:s and laptop computers, to cell phones and smaller devices embedded in our homes, in cars and in the environment around us. The Internet is also an infrastructure that supports a diversity of applications like the web, mail, file sharing, telephony, radio, video and TV distribution, games, banking and commerce of many kinds; and where new applications constantly are developed and deployed.

Taken as a whole the Internet is a very complex system. To get some structure in this, one could notice two things: First, the Internet is a network of networks. It consists of a large number of smaller and independently managed networks. Secondly, the protocols that define how Internet communication is done are structured into layers with different functionality.

The Internet is a network of interconnected heterogeneous networks of different sizes, different capacities, and under different administrations. The hundreds of thousands of networks that constitute the Internet are connected together in a loose hierarchy. At the top there are a small number of tier-1 operators with large international high-capacity networks. The tier-1 networks directly connect to each other and the operators have peering agreements that allow data to flow between the networks without charging each other for the

data transmitted. A tier-2 network is typically a regional or national network. It can have peering agreements with other tier-2 networks to exchange traffic but it is also a customer to one or more tier-1 operators and need to buy transit to reach some parts of the Internet. At the bottom of the network hierarchy are the access networks that connects the end hosts to the Internet. These are typically local telephone companies, university or company networks that in turn are customers to upper-tier networks to be able to communicate worldwide.

The Internet protocols are arranged in layers, each having a different responsibility. The TCP/IP protocol suite used for Internet communication follows a 4-layer model with different protocols at these layers. Figure 1.1 illustrates the layered architecture with example protocols at the different layers. At the top is the application layer which includes many different protocols for

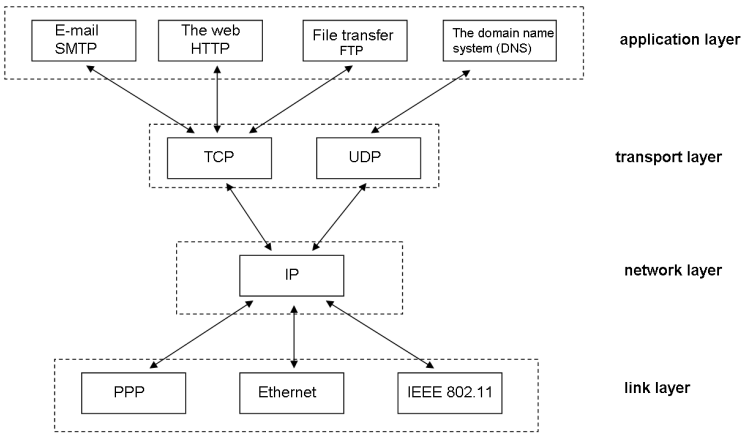


Figure 1.1: Layers with example protocols

handling the details of particular applications. For instance the Simple Mail Transfer Protocol (SMTP) for transferring electronic mail messages and the HyperText Transfer Protocol (HTTP) for requests and transfers of web pages.

The transport layer provides a flow of data between two hosts. TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) are the two predominant transport layer protocols. TCP provides a connection-oriented, reliable, byte stream service to the application layer. It keeps track of the packets sent and retransmits packets that are lost in the network. TCP also provides flow control and congestion control that adapts the transmission rate to what

the endhosts and the network can handle.

The network layer contains the Internet protocol (IP) that defines the IP packet as the unit of information passed across the Internet. The network layer handles the movement of packets around the network using routing and forwarding mechanisms. Routing determines the paths that packets should take through the networks from sender to receiver. Routing protocols are used to create forwarding tables. Routers then use the tables to forward packets towards the receiver based on the destination IP-address in the packet. The network layer and the IP protocol interconnect the many different networks that constitute the Internet and provide communication across many kinds of link layer technologies.

1.2 Internet traffic characteristics

How to best model and describe Internet traffic is still an open area of research. The traffic characteristics depend on when and where on the Internet the traffic is investigated. The traffic behaviour in a large backbone network differ from that in a small company network, and the traffic characteristics changes with new applications, new types of networks and with changing user behaviour.

Ten years ago, measurements on the Internet backbone showed that 70-75% of the traffic was web traffic [3]. Since then the total traffic volumes have increased a lot, the share of web traffic is still high in many networks [4, 5, 6] but now often file sharing is the application that dominates the traffic [7, 8]. Also, TV and video distribution over IP are becoming widespread and produce increasing traffic volumes.

Figure 1.2 shows examples of Internet traffic behaviour. On short time scales up to seconds the traffic is very bursty and on longer timescales there are often predictable daily and weekly cycles and in between there can be unpredictable shifts and changes in traffic demand.

1.3 Internet traffic management

Internet traffic management means handling the traffic situation in the networks; avoiding congestion and making good use of available network resources.

Traffic management involves both the end hosts and the network operators. It involves the end hosts in that they for many applications run TCP congestion control and adapt the send rate to what the network can handle. TCP increases

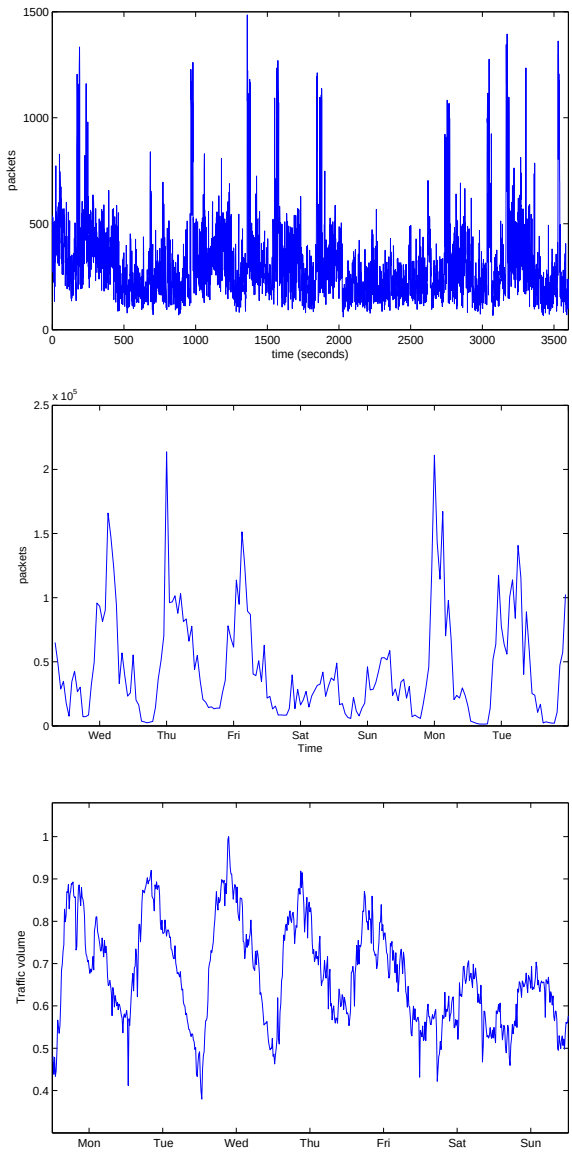


Figure 1.2: Example of Internet traffic behaviour. Top: Packets per second during one hour in a company network. Middle: Web traffic per hour during one week in a company network. The traffic data is described in paper A [9]. Bottom: Total traffic in a backbone network during one week (normalised). Traffic data from the Geant network [10].

the send rate to find out the available network capacity. When a packet is lost this is interpreted as network congestion and the transmission rate is decreased. From a network operator perspective traffic management involves monitoring and understanding the traffic behaviour in the network. It also includes traffic engineering where the routing of traffic through the network is adapted to the current traffic situation.

For network operators it is important to manage the traffic situation in the network and meet service level agreements (SLAs) made with their customers. The traffic demands in a network may fluctuate and changes over time. Traffic engineering mechanisms can then be used to adapt to the changes in traffic demand and distribute traffic to benefit from available network resources. Today, the main alternative for traffic engineering within an IP network is to use different methods for setting the link costs (and so decide upon the shortest paths) in the routing protocols OSPF (Open Shortest Path First) and IS-IS (Intermediate System to Intermediate System). These are both link-state protocols and the routing decisions are based on link costs and a shortest (least-cost) path calculation. With the equal-cost multi-path (ECMP) extension to the routing protocols the traffic can also be distributed over several paths that have the same cost. These routing protocols were designed to be simple and robust rather than to optimise the resource usage. They do not by themselves consider network utilisation and do not always make good use of network resources. The traffic is routed on the shortest path through the network even if the shortest path is overloaded and there exist alternative paths. It is up to the operator to find a set of link costs that is best suited for the current traffic situation and that avoids congestion in the network.

Network operators today have different strategies for coping with traffic variability: ranging from just over-dimensioning network capacity a lot, to occasionally tuning the link costs in OSPF to postpone upgrades of network equipment, to more active use of traffic monitoring and traffic engineering mechanisms to manage the traffic situation.

Chapter 2

Research Issues

This thesis presents work on three aspects of Internet traffic management: traffic engineering, web traffic modelling, and bandwidth allocation to TCP flows. The areas all have in common the need to understand and handle Internet traffic behaviour. For web traffic modelling the goal of the work itself is to understand traffic behaviour and to be able to generate realistic traffic in simulations and lab experiments. For traffic engineering and bandwidth allocation to TCP flows the purpose is to develop methods to control and steer the traffic.

2.1 Characterizing web traffic and generating synthetic traffic

Surfing the web is one of the most popular Internet applications. Ten years ago, measurements on the Internet backbone showed that 70-75% of the traffic was web traffic [3]. Since then the total traffic volumes have increased a lot, other applications like file sharing now often dominates the traffic, but the share of web traffic is still high in many networks [4, 5, 6]. It is therefore important to measure and model web traffic in order to understand and manage the behaviour of aggregated Internet traffic, and to be able to generate realistic traffic in simulations and lab experiments.

The research issues are to collect web traffic, extract and model the important characteristics from the collected data, and from the model generate realistic synthetic traffic.

A direct method to collect information about web traffic is to use a packet

trace. But we do not want to model the number, size and inter-arrival times of packets. These packet level characteristics depend on the TCP flow control and congestion control algorithms and reflects the conditions in the network at the time the trace was taken. To be able to generate realistic traffic for different levels of network utilisation we want a higher level model on top of TCP.

In paper A we present a simple model of web client traffic. We start with a packet trace of web traffic and use a heuristic method to identify when a user clicks on a link to retrieve the next web page. From this we derive empirical probability distributions describing session lengths, time between user clicks, and the amount of data that is transferred due to a single user click. Using these probability distributions we implement and evaluate a web-client traffic generator.

2.2 Dynamic allocation of bandwidth to TCP flows

TCP is the predominant Internet transport protocol. TCP is used by many popular applications including the web and it is used for transporting approximately 80-90% of the Internet traffic [4, 6, 7, 5].

TCP is a reliable end-to-end transport protocol that adapts its rate to the available capacity. Network technologies such as Dynamic synchronous Transfer Mode (DTM) provides channels with dynamically adjustable capacity. The issue here is to adaptively allocate bandwidth to TCP flows, when both TCP and the bandwidth allocation scheme can react to changes in the network load.

In paper B, we do a simulation study in ns-2 and investigate the behaviour of a bandwidth allocation scheme, its effect on TCP flows and on a network that can vary its capacity.

2.3 Robust traffic engineering

The objective of traffic engineering is to avoid congestion in the network and to make better use of available resources by adapting the routing to the current traffic situation. The main challenge for traffic engineering is to cope with the dynamics of traffic demands and topology. Traffic is often bursty and there can be unpredictable changes and shifts in traffic demand, for instance due to hotspots and flash crowds, or because a link goes down, there are changes in the inter-domain routing, or because traffic in an overlay is re-directed. For future networks more variability in traffic demands is also expected due to mobility of nodes and networks and more dynamic on-demand service level agreements.

The traffic variability means that, even if we could measure the current traffic situation exactly, it would not always correctly predict the near future traffic situation. Traffic engineering mechanisms need to be robust and able to handle traffic variability and uncertainties in input traffic data.

The papers C, D and E in this thesis cover different aspects of robust traffic engineering.

2.4 Contributions

The contributions of the web traffic work are the heuristics for detecting user clicks in a packet trace, a simple empirical web client model (describing session lengths, time between user clicks, and the amount of data that is transferred due to a single user click), and a realistic web client traffic generator.

For studying dynamic allocation of bandwidth to TCP flows we implemented a simulation environment in ns-2. The contribution of this work is the performance evaluation of an estimation algorithm, which measures the rate of TCP flows and allocates capacity on a DTM network.

We show that the measurement-based bandwidth allocator usually works well for TCP flows but we also identify a scenario where packet loss makes the feedback mechanisms interact in an unfavourable way and where the bandwidth allocator could be improved. If the bandwidth allocator somehow fails to assign enough capacity and packets are dropped then TCP decreases its rate. The measurement-based estimator then decreases rather than increases the allocated bandwidth.

For robust traffic engineering we propose l-balanced routings as a way for an operator to handle traffic variability and uncertainties in input traffic data. An l-balanced solution routes the traffic on the shortest paths possible but makes sure that no link is utilised to more than a given level l. The contributions are an l-balanced routing algorithm based on multi-commodity flow optimisation and a heuristic search method for finding l-balanced weight settings for the legacy routing protocols OSPF and IS-IS.

L-balanced routing gives the operator possibility to apply simple rules of thumb for controlling the maximum link utilisation and control the amount of spare capacity needed to handle sudden traffic variations. It gives more controlled traffic levels than other cost functions and more efficient routing for low traffic loads when there is no need to spread traffic over longer paths.

Chapter 3

Summary of the Papers and Their Contributions

The thesis is a collection of five papers. Paper A about web traffic characterisation, paper B about bandwidth allocation to TCP flows, and paper C-E on different aspects of robust traffic engineering. Paper A, B and C are all published at refereed international conferences. Paper D was published at the Swedish National Computer Networking Workshop and paper E is submitted for publication.

3.1 Paper A

Using Empirical Distributions to Characterize Web Client Traffic and to Generate Synthetic Traffic. Henrik Abrahamsson and Bengt Ahlgren. In *Proceedings of IEEE Globecom:Global Internet, San Francisco, USA, November 2000.*

Summary:

This paper presents a simple model of web client traffic. The paper describes how the model is derived, some characteristics of web traffic, and how the model is used to implement a traffic generator.

A packet trace with HTTP traffic data captured at SICS is analysed to obtain the traffic characteristics of web clients. A heuristic method is used to identify user clicks in the packet trace and from this empirical probability distributions are derived describing session lengths, time between user clicks, and

the amount of data that is transferred due to a single user click. Using these probability distributions a web-client traffic generator is implemented and evaluated.

Contribution:
The contributions of this work are the heuristics for detecting user clicks in a packet trace (based on the work by Mah [11]), a simple empirical web client model and a realistic web client traffic generator.

My contribution:
I performed the work and wrote the paper under supervision of Bengt Ahlgren. I analysed the packet traces, came up with the heuristic method used, and derived the empirical probability distributions that make up the model. I implemented and evaluated the traffic generator. I wrote most of the paper except for parts of introduction and conclusions. I later also implemented the traffic generator in the ns-2 network simulator and used it for network dimensioning in an industrial project with Teracom AB.

3.2 Paper B

TCP over High Speed Variable Capacity Links: A Simulation Study for Bandwidth Allocation. Henrik Abrahamsson, Olof Hagsand and Ian Marsh. In *Proceedings of Protocols for High-Speed Networks (PfHSN 2002), Berlin, Germany, April 2002.*

Summary:
This paper presents a simulation study of bandwidth allocation to TCP flows. Dynamic synchronous Transfer Mode (DTM) is a gigabit network technology that provides channels with dynamically adjustable capacity. TCP is a reliable end-to-end transport protocol that adapts its rate to the available capacity. Both TCP and the DTM bandwidth can react to changes in the network load, creating a complex system with inter-dependent feedback mechanisms.

In this work we create a simulation environment using ns-2. We investigate the behaviour of a bandwidth allocation scheme, its effect on TCP flows and on a network that can vary its capacity. The results indicate that the bandwidth allocation scheme usually works well for TCP flows. But the paper also highlight a scenario where packet loss make the feedback mechanisms interact in an unfavourable way and where the allocation scheme could be improved.

Contribution:
The contribution of this work is an assessment of a bandwidth allocation scheme

for TCP flows on variable capacity technologies. For evaluating the bandwidth allocator we implemented a simulation environment in ns-2.
My contribution:
This work was done in collaboration with Ian Marsh and Olof Hagsand. Together with Ian Marsh, I implemented the ns-2 simulation environment, planned and ran the experiments and analysed the results. I co-authored the paper.

3.3 Paper C

A Multi Path Routing Algorithm for IP Networks Based on Flow Optimisation H. Abrahamsson, J. Alonso, B. Ahlgren, A. Andersson and P. Kreuger. In *Proceedings of Third COST 263 International Workshop on Quality of Future Internet Services(QoFIS 2002)*, Zurich, Switzerland, October 2002.
Summary:
Intra-domain routing in the Internet normally uses a single shortest path to forward packets towards a specific destination with no knowledge of traffic demand. We present an intra-domain routing algorithm based on multi-commodity flow optimisation which enable load sensitive forwarding over multiple paths. It is neither constrained by weight-tuning of legacy routing protocols, such as OSPF, nor requires a totally new forwarding mechanism, such as MPLS. These characteristics are accomplished by aggregating the traffic flows destined for the same egress into one commodity in the optimisation and using a hash based forwarding mechanism. The aggregation also results in a reduction of computational complexity which makes the algorithm feasible for on-line load balancing. Another contribution is the optimisation objective function which allows precise tuning of the tradeoff between load balancing and total network efficiency.
Contribution:
There are two contributions in this paper: the modelling of the problem as an optimisation problem, and the definition of an optimisation objective function for *l*-balanced solutions.

In the modelling of the optimisation problem we aggregate all traffic destined for a certain egress into one commodity in a multi-commodity flow optimisation. It is this definition of a commodity that both makes the computation tractable, and the forwarding simple.

L-balanced solutions allows the network operator to choose a maximum desired link utilisation level. The optimisation will then find the most efficient solution, if it exists, satisfying the link level constraint. Our objective function

thus enables the operator to control the trade-off between minimising the network utilisation and balancing load over multiple paths.
My contribution:
This is joint work with Bengt Ahlgren, Juan Alonso, Anders Gunnar and Per Kreuger. Juan Alonso did most of the mathematical work for this paper. In discussion with Juan I contributed to the idea of only looking at the destination of the traffic when formulating the optimisation problem. I co-authored the paper.

3.4 Paper D

Traffic Engineering in Ambient Networks: Challenges and Approaches
H. Abrahamsson and A. Gunnar. In *Proceedings of Second Swedish National Computer Networking Workshop (SNCNW)*, 2004, Karlstad, Sweden.
Summary:
This paper identifies the requirements and challenges for traffic engineering in a dynamic environment. We give a short introduction to the Ambient Networks project which aims to provide a novel mobile communication platform beyond 3G. Further, a framework for classification of traffic engineering methods is introduced to facilitate the analysis and identification of challenges and alternatives for traffic engineering in Ambient Networks.
Contribution:
The contribution of this paper is in the identification of and reasoning about requirements, challenges and alternatives for traffic engineering in a dynamic environment.
My contribution:
I did this work in cooperation with Anders Gunnar. I wrote about half of the paper.

3.5 Paper E

Robust Traffic Engineering using L-balanced Weight-Settings in OSPF/ISIS
H. Abrahamsson and M. Björkman. Submitted for publication September 2008.
Summary:
The focus of this work is on robust traffic engineering for the legacy routing protocols OSPF and IS-IS. The idea is to use the L-balanced solutions proposed in paper C to make sure that there are enough spare capacity on all links

to handle sudden hotspots and traffic shifts. Search heuristics are used to find the set of weights that avoid loading any link to more than L and the resulting routings are evaluated using real topologies and traffic scenarios.

Contribution:

The contributions are the idea of *l*-balanced weight-settings for robust traffic engineering, the search heuristics for finding such weight-settings, and the evaluation of how different cost functions (including *l*-balanced) manage to handle faults in input traffic data due to traffic hotspots.

My contribution:

The idea of using the *l*-balanced solution for robust weight-settings was mine. I implemented the search heuristics and did the evaluations and wrote most of the paper.

Chapter 4

Related Work

4.1 Characterizing web traffic and generating synthetic traffic

In paper A from 2000 we model web client traffic and implement a traffic generator with the purpose of creating realistic background traffic in simulations and lab experiments.

Two approaches can be used to generate network traffic that imitates real web traffic. The first is simply to replay packet traces of real web traffic. But, because of TCP's congestion control the timing of packets in a trace reflects the condition in the network when the trace was taken and this timing would not be the same in another context. The better alternative, as argued by Paxson and Floyd [12], is to gather information about and model those aspects of the web traffic that one believe is most important and from this model generate traffic.

4.1.1 Methods for collecting information about web traffic

To get information about web traffic three different approaches have been widely used: server logs, client logs and packet traces. Server logs cannot easily be used to describe the client side since a client usually accesses many different web servers.

To capture the client accesses between multiple servers, client logs can be used. This approach requires that browsers can log their requests, that the source code for the browser is available so that logging can be added, or

that some other way to log the clients behaviour is available. Catledge and Pitkow [13], Cunha *et al.* [14] and Crovella and Bestavros [15] used instrumented versions of the Mosaic web browser and in Barford *et al.* [16] HTTP proxies were used to track all documents referenced by unmodified Netscape Navigator clients. Later work that use this method include the tool Carena [17] which together with the Mozilla web browser can be used to capture and replay browsing sessions.

The third approach of gathering data, and the method we use in [9], is to analyse packet traces taken from a subnet carrying HTTP traffic. This method was used by Stevens [18] to analyse the traffic arriving at a server, and by Mah [11] to model the client side of the HTTP traffic. A further step is taken by Anja Feldmann [19] when extracting full HTTP level as well as TCP level traces via packet monitoring. The method of investigating packet traces is also used by Choi and Limb [20], Smith *et al.* [21], Molina *et al.* [22], and Tran *et al.* [23]. Recently Cao *et al.* [24], Weigle *et al.* [25] and Vishwanath and Vahdat [26] all use packet traces from a single point in the network to characterise and generate traffic.

Simpson *et al.* [27] use the NETI@home [28] software to collect statistics from end-systems. The approach here is to sniff packets sent to and from the host (that is running the software) and infer statistics based on these observed packets. The dataset used in [27] to investigate web traffic includes 1700 users in 28 nations.

4.1.2 Investigated characteristics of web traffic

Different studies of web traffic look at different properties of the traffic. The two most common characteristics to investigate (and what we modelled in paper A) is user OFF times and response sizes. User OFF time (sometimes called user view time or user think time) is the time from when a download of a web page is completed to the next request. Response sizes are the number of bytes that is downloaded in response to a web request.

Mah [11] investigate user OFF times and response sizes but also request sizes, number of files per page, number of consecutive document from the same server and web server popularity. Barford and Crovella [16] look at request and response sizes, number of files per page, user OFF times, the popularity of web pages, and temporal locality (meaning the likelihood that once a file has been requested, it will be requested again in the near future). Smith *et al.* [21] analyse request and response sizes. Choi and Limb [20] look at web request and response sizes, number of objects per page, user viewing time and web

page caching. Molina *et al.* [22] investigate response sizes, user think times, and also number of TCP connections used to download a page. Simpson *et al.* [27] examine bytes sent and bytes received, user think time, and the frequency distribution of contacting specific destinations.

4.1.3 Web traffic generators

One of the most well-known web traffic generators is SURGE [16]. The tool generates traffic matching distributions of request and response sizes, number of files per page, user OFF times, the popularity of web pages, and temporal locality. Also, the web models described by Mah [11] and Choi and Limb [20] has been used to generate synthetic traffic.

More recent work on traffic generators include the tools PackMime-HTTP [24], Tmix [25] and Swing [26].

4.2 Dynamic allocation of bandwidth to TCP flows

Lundqvist [29] evaluates different algorithms for bandwidth allocation for DTM channels transporting IP traffic. The algorithms were assessed with respect to throughput, delay and bandwidth changes per second. TCP rate adjustment is done by placing the incoming packets into a buffer and adding and removing slots if the level of the buffer exceeds continuously maintained threshold values. He concludes that adaptive strategies are recommended for TCP, however too frequent changes can be undesirable in a DTM network due to the processing cost. The main conclusion from this work is that the choice of algorithm can play a significant role in the performance. This work is similar to ours in that the goal is a slot allocation for TCP traffic over DTM. It differs from ours in that we measure the rate of each TCP flow, whilst he looks at the outgoing buffer length as a sign to increase or decrease the number of slots.

Krishnan and Sterbenz investigate TCP over load-reactive links [30, 31]. They use a hysteresis control mechanism for capacity allocation. Buffer levels are monitored (as in [29]) and if the occupancy is greater than a threshold the capacity is increased and vice versa. Their scheme is dependent on keeping buffers occupied all the time, otherwise the link capacity will fall and hence the throughput. A single TCP flow is simulated and the authors state that the control parameters should be carefully chosen. Poor parameter choice can have the opposite effect, resulting in TCP not being able to operate. The work resembles ours in that a method is presented to react to network load and allocate

bandwidth for TCP accordingly. It differs in that we measure the throughput of individual flows and allocate bandwidth from this measurements, where they use the buffer length as a measure of the load.

Clark and Fang propose a framework for allocating bandwidth to different users during congestion [32]. The focus of the work is TCP bulk-data transfers. The authors attempt to keep TCP flows in congestion avoidance in the best case, and fast recovery phase in the worst case, by avoiding dropping several packets of the same flow in the same RTT. The work resembles ours in that they attempt to allocate bandwidth between different flows in a fair manner. It differs from ours in that we assume that the network can change its offered bandwidth and we focus on maximising TCP throughput, rather than trying to maintain a TCP state in the face of adverse network conditions. In addition, we allocate bandwidth to flows not only when the network is congested but also in normal situations as well.

Comprehensive studies have been done related to the performance of TCP on ATM networks [33, 34]. The main conclusions of the works are similar, the traffic classes of ATM are poorly suited to the bursty needs of TCP, due to the traffic contracts needed by ATM classes. The conclusion in Bonaventure [33] is that the complexity of choosing traffic parameters for ABR is not in proportion to the benefits of carrying TCP/IP traffic. The CBR class is too simple for TCP, as only the peak rate is specified.

TCP can also interact with routing decisions and traffic engineering done by operators on the network layer. TCP and traffic engineering work independently of each other, but both try to avoid congestion and make good use of network resources.

Gao *et.al* [35] study the interaction between TCP and a route control mechanism for multihomed networks that selects egress link based on performance measurements. They show that the route selection often can improve TCP throughput and that the two mechanism interact well given that the route control react on longer timescales than TCP.

Anderson and Anderson [36] study the interaction between the feedback mechanisms of adaptive routing and congestion control. They argue that adaptive routing can be designed to be stable in conjunction with congestion control.

He *et.al* [37, 38] model the interaction between TCP congestion control and traffic engineering in a network. They show through simulation that the mechanisms are stable and work effectively together.

4.3 Traffic engineering in IP networks

Traffic engineering by finding a suitable set of weights in OSPF/IS-IS is a well studied area of research and it is described in recent textbooks in the area [39, 40]. When we in paper E revisit the weight setting approach to traffic engineering we are most inspired by the pioneering works by Fortz and Thorup [41, 42] and Ramakrishnan and Rodrigues [43], in that we use a piece-wise linear cost function and search heuristics to find suitable weight settings.

Several studies [41, 44, 45, 46] have shown that even though we limit the routing of traffic to what can be achieved with weight-based ECMP shortest paths, and not necessarily the optimal weights but those found by search heuristics, it often comes close to the optimal routing for real network scenarios. How the traffic is distributed in the network very much depends on the objectives, usually expressed as a cost function, in the optimisation. An often proposed objective function is described by Fortz and Thorup [41]. Here the sum of the cost over all links is considered and a piece-wise linear increasing cost function is applied to the flow on each link. The basic idea is that it should be cheap to use a link with small utilisation while using a link that approaches 100% utilisation should be heavily penalised. The *l*-balanced cost function used in paper C and E is similar in that it uses a piecewise linear cost function to obtain desirable solutions. Additionally, it gives the operator the opportunity to set the maximum wanted link utilisation. Cost functions for traffic engineering is further investigated by Balon *et.al* [47]

Paper E add to existing work on weight settings by focusing on robustness and the objective of achieving a controlled spare capacity for handling unpredictable traffic shifts. For robust traffic engineering much of the focus is on handling multiple traffic matrices and traffic scenarios [42, 48, 49, 50, 51] and handling the trade-off between optimising for the common case or for the worst case. Nucci *et.al* [52] investigate link weight assignments that take into account SLA requirements and link failures.

Xu *et.al* [53] describe a method to jointly solve the flow optimisation and the link-weight approximation using a single formulation resulting in a more efficient computation. Their method can also direct traffic over non-shortest paths with arbitrary percentages. Their results should also be directly applicable to our problem of providing robustness to changes, by just substituting their piece-wise linear cost function with our cost function. In a continuation on this work Xu *et.al* [54] propose a new link-state routing protocol. The protocol splits traffic over multiple paths with an exponential penalty on longer paths and achieves optimal traffic engineering while retaining the simplicity of

hop-by-hop forwarding.

The advantage of optimising the weights in OSPF and IS-IS is of course easy deployment of the traffic engineering mechanism. However, the disadvantage is the difficulties and constraints imposed by using legacy routing. Much of the research also start to take a clean slate approach, not limited to what can be achieved with todays protocols and argue that future protocols should be designed with optimisation and manageability in mind from the beginning [55]. The general problem of finding the best way to route traffic through a network can be mathematically formulated as a multi-commodity flow (MCF) optimisation problem. In paper C we present a routing algorithm based of multi-commodity flow optimisation. By aggregating the traffic flows destined for the same egress into one commodity in the optimisation we reduce the computational complexity. The same approach is used by Fu *et.al* [56]. They use multi-commodity flow optimisation for centralised traffic engineering combined with a scheme to quickly recompute routing paths when the topology changes. MCF optimisation is also used by many other research groups to address traffic engineering problems including [41, 45, 57]. See also the book by Pioro and Medhi [39] and references therein.

Most traffic engineering methods (including the weight-setting and optimisation methods described above) need as input a traffic matrix describing the traffic demand between each pair of nodes in the network. With this network-wide information the routing can often come close to optimal. But the drawback is that it is difficult to react quickly to changes without too much traffic overhead. With local traffic engineering on the other hand the individual nodes can quickly react to changes in traffic but can possibly create routing loops and overload elsewhere in the network. An attempt to localise and distribute the routing decisions is Adaptive Multi-path routing (AMP) [58]. In AMP information on the traffic situation on links is only distributed to the immediate neighbours of each router. Hence, AMP relies on local information in neighbouring routers to calculate next hop towards the destination. The Multi-Path Routing with Dynamic Variance (MRDV) [59] combined with a Loop Avoidance Protocol (LAP) [60] is another approach to localised traffic engineering. In this approach no load information is exchanged between routers. Instead the cost of each path towards the destination is weighted by a variance factor which reflect load on the next hop. Hence, traffic is shifted from heavily loaded links to links with less load. A related approach is introduced by Vutukury *et.al* [61]. Here the routing decision is divided into two steps. First, multiple loop-free paths are established using long term delay information. In the second step the routing parameters along the precomputed paths are adjusted using

only local short-term delay information.

Local versus network-wide traffic information and novel routing mechanisms versus optimising legacy routing are two ways of categorising traffic engineering methods. A detailed taxonomy of traffic engineering methods can be found in RFC 3272 [62] .

Chapter 5

Conclusions

Internet traffic volumes continue to grow at a great rate, now pushed on by video and TV distribution in the networks. Increasing traffic volumes and the introduction of delay and loss sensitive services makes it crucial for operators to understand and manage the traffic situation in the network. More traffic also necessitate upgrades of network equipment and new investments for operators, and keep up-to-date the question of over-dimensioning network capacity versus using traffic engineering mechanisms for better handling the traffic.

This thesis approaches Internet traffic management on different levels: we investigate web client behaviour and web traffic characteristics on the application layer; we study bandwidth allocation to TCP flows on the transport layer; and traffic engineering mechanisms for adapting the routing to the current traffic situation on the network layer.

The thesis presents a simple model of web client traffic and based on this model a traffic generator used for generating realistic traffic in simulations and lab experiments. The traffic generator has also been implemented in the ns-2 simulator and used in an industrial project for dimensioning and investigation of how many web users a given network can handle.

This work, presented in paper A, was done in the year 2000. Since then many new applications have appeared. Today, even though web traffic still contribute to a large part of the total traffic in some networks, one would also need to model other applications like file sharing to get a representative mixture of traffic.

Also, the web itself has developed a lot and the use of the web has changed during the last eight years. Web surfing is no longer a sequential transfer of

static webpages consisting of a html document and a few images. Web pages today are often much more complex, dynamic and interactive with videos and advertisements. Downloading a commercial web page can involve communicating with ten different servers including content delivery networks and advertising companies. The distribution of transfer sizes would look different today.

Web users also often have many concurrent tabs and web browser windows running at the same time; for instance having a music video running in the background (over port 80), and at the same time reading email and clicking on links on other web pages. This means that the simple heuristic for detecting user clicks in paper A no longer would work; and it would be difficult to model the user behaviour using only packet traces.

If the purpose is to generate realistic traffic based on packet traces then an alternative [25, 26] is to have a more abstract model of transfers above TCP. This type of model can include off times and response sizes but does not necessarily capture the high level user and application behaviour; i.e how often the user clicks on links or how the application involves communication with many different server.

The thesis also study dynamic allocation of bandwidth to TCP flows. We evaluate an estimation algorithm, which measures the rate of TCP flows and allocates capacity to channels in a DTM network. We use simulation to study the interaction between the TCP congestion control mechanism and the bandwidth allocation scheme which both react to changes in network load.

Paper B shows that the measurement-based bandwidth allocator usually works well for TCP flows but it also highlight a scenario where packet loss make the feedback mechanisms interact in an unfavourable way.

The traffic demands in a network change over time and there can be unpredictable changes and shifts, for instance due to hotspots, or because a link goes down, or because traffic in an overlay is re-directed. For future networks (as discussed in paper D) more variability in traffic demands is also expected due to mobility of nodes and networks and more dynamic on-demand service level agreements (SLA:s). This means that a network operator can not rely only on long-term network planning and dimensioning that are done when the network is first built. Robust traffic engineering mechanisms are needed that can adapt to changes in traffic demand and distribute traffic to benefit from available resources.

This thesis propose l-balanced routings as a way for an operator to handle traffic variability and uncertainties in input traffic data. An l-balanced routing algorithm based on multi-commodity flow optimisation was presented in pa-

per C. A heuristic search method for finding *l*-balanced weight settings for the legacy routing protocols OSPF and IS-IS was presented in paper E.

L-balanced routing gives the operator possibility to apply simple rules of thumb for controlling the maximum link utilisation and control the amount of spare capacity needed to handle sudden traffic variations. It gives more controlled traffic levels than other cost functions and more efficient routing for low traffic loads when there is no need to spread traffic over longer paths. Paper E shows that the search and the resulting weight settings work well in real network scenarios.

Bibliography

[1] Internet domain survey host count. On–line: http://www.isc.org. Internet Systems Consortium.

[2] Internet world stats. http://www.internetworldstats.com/stats.htm.

[3] G. J. Miller K. Thompson and R. Wilder. Wide-area traffic patterns and characteristics (extended version). *IEEE Network*, 11(6):10–23, 1997.

[4] Richard Nelson, Daniel Lawson, and Perry Lorier. Analysis of long duration traces. *SIGCOMM Computer Communication Review*, 35(1):45–52, 2005.

[5] Chuck Fraleigh, Sue Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and Christophe Diot. Packet-Level Traffic Measurements from the Sprint IP Backbone. *IEEE Network*, 2003.

[6] Abilene network usage report.
On–line:http://netflow.internet2.edu/weekly/.

[7] CAIDA Internet Data – Realtime Monitors.
On–line:http://www.caida.org/data/realtime/index.xml.

[8] Wolfgang John, Sven Tafvelin, and Tomas Olovsson. Trends And Differences in Connection Behavior within Classes of Internet Backbone traffic. In *Prooceedings of the 9th Passive and Active Measurement Conference*, Cleveland, USA, 2008.

[9] Henrik Abrahamsson and Bengt Ahlgren. Using empirical distributions to characterize web client traffic and to generate synthetic traffic. In *Proceedings of IEEE Globecom: Global Internet*, San Francisco, USA, November 2000.

[10] S. Uhlig, B. Quoitin, S. Balon, and J. Lepropre. Providing public intradomain traffic matrices to the research community. *ACM SIGCOMM Computer Communication Review*, 36(1), January 2006.

[11] B. A. Mah. An empirical model of HTTP network traffic. In *Proceedings of INFOCOM '97*, Kobe, Japan, April 1997.

[12] V. Paxson and S. Floyd. Why we don't know how to simulate the Internet. In *Proceedings of the 1997 Winter Simulation Conference*, Atlanta, GA, USA, 1997.

[13] L. D. Catledge and J. E. Pitkow. Characterizing browsing strategies in the World Wide Web. In *Proceedings of the third International World Wide Web Conference on technology, tools and applications*, Darmstadt, Germany, April 1995.

[14] C. R. Cunha, A. Bestavros, and M. E. Crovella. Characteristics of WWW client-based traces. Technical Report Technical Report BU-CS-95-010, Computer Science Department, Boston University, July 1995.

[15] M. E. Crovella and A. Bestavros. Self-similarity in World Wide Web traffic: Evidence and possible causes. *IEEE/ACM Transactions on Networking*, 5(6):835–846, 1997.

[16] P. Barford, A. Bestavros, A. Bradley, and M. Crovella. Changes in Web client access patterns: Characteristics and caching implications. *World Wide Web*, 2:15–28, 1999. Special issue on Characterization and Performance Evaluation.

[17] I. J. Nino, B. de la Ossa, J.A Gil, J. Sahuquillo, and A. Pont. CARENA: A tool to capture and replay web navigation sessions. In *Proceedings of IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Services (E2EMON 2005)*, Nice, France, 2005.

[18] W. R. Stevens. *TCP/IP illustrated, Vol. 3, TCP for Transactions, HTTP, NNTP, and the UNIX Domain Protocols*. Addison-Wesley, 1996.

[19] A. Feldmann. BLT: Bi-Layer Tracing of HTTP and TCP/IP. In *Proceedings of 9th International World Wide Web Conference*, Amsterdam, May 2000.

[20] H. Choi and J. O. Limb. A behavioral model of web traffic. In *Proceedings of the Seventh International Conference on Network Protocols (ICNP'99)*, Toronto, Canada, 1999.

[21] F. D. Smith, F. H. Campos, K. Jeffay, and D. Ott. What TCP/IP protocol headers can tell us about the web. In *Proceedings of ACM SIGMETRICS 2001*, Cambridge, Massachusetts, USA, 2001.

[22] M. Molina, P. Castelli, and G. Foddis. Web traffic modelling exploiting TCP connections temporal clustering through HTML-REDUCE. *IEEE Network*, 14, 2000.

[23] D. N. Tran, W. T. Ooi, and Y. C. Tay. SAX: A tool for studying congestion-induced surfer behavior. In *Proceedings of Passive and Active Measurement Workshop (PAM 2006)*, Adelaide, Australia, March 2006.

[24] J. Cao, W. S. Cleveland, Y. Gao, K. Jeffay, F. D. Smith, and M. Weigle. Stochastic models for generating synthetic HTTP source traffic. In *Proceedings of IEEE INFOCOM*, 2004.

[25] M.C. Weigle, P. Adurthi, F. Hernandez-Campos, K. Jeffay, and F.D. Smith. Tmix: A tool for generating realistic TCP application workloads in ns-2. *ACM SIGCOMM Computer Communication Review*, 36(3), July 2006.

[26] K. V. Vishwanath and A. Vahdat. Realistic and responsive network traffic generation. In *Proceedings of ACM SIGCOMM'06*, Pisa, Italy, September 2006.

[27] C.H. Simpson, D. Reddy, and G.F. Riley. Empirical models of TCP and UDP end-user network traffic from NETI@home data analysis. In *Prooceedings of 20th IEEE/ACM/SCS Workshop on Principles of Advanced and Distributed Simulation (PADS 2006)*, Singapore, May 2006.

[28] NETI@home. On–line: http://neti.gatech.edu. Georgia Institute of Technology.

[29] Henrik Lundqvist. Performance evaluation for IP over DTM. Master's thesis, Linköping University, Linköping, 1998.

[30] Rajesh Krishnan and James Sterbenz. TCP over load-reactive links. In *International Conference on Network Protocols (ICNP)*, 2001.

[31] Rajesh Krishnan. TCP over load-reactive links. Technical Memorandum 1281, BBN, February 2001.

[32] D. Clark and W. Fang. Explicit allocation of best-effort packet delivery service. *IEEE/ACM Transactions on Networking*, 6(4), August 1998.

[33] O. Bonaventure. *Integration of ATM under TCP/IP to provide services with minimum guaranteed bandwidth*. PhD thesis, University of Liege, 1998.

[34] E. Chan, V. Lee, and J. Ng. On the performance of bandwidth allocation strategies for interconnecting ATM and connectionless networks. *ACM Computer Communications Review*, 26(1):29–38, January 1996.

[35] R. Gao, D. Blair, C. Dovrolis, M. Morrow, and E. Zegura. Interactions of Intelligent Route Control with TCP Congestion Control. In *Proceedings of IFIP Networking*, May 2007.

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

[37] Jiayue He, Mung Chiang, and Jennifer Rexford. Can Congestion Control and Traffic Engineering Be at Odds? In *Proceedings of IEEE GLOBECOM*, San Francisco, USA, December 2006.

[38] Jiayue He, Ma'ayan Bresler, Mung Chiang, and Jennifer Rexford. Towards Robust Multi-layer Traffic Engineering: Optimization of Congestion Control and Routing. *IEEE Journal on Selected Areas in Communications*, June 2007.

[39] M. Pióro and D. Medhi. *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann, 2004.

[40] J. Rexford. Route optimization in IP networks. In Mauricio G.C. Resende and Panos M. Pardalos, editors, *Handbook of Optimization in Telecommunications*. Springer Science+Business Media, 2006.

[41] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *Proceedings IEEE INFOCOM 2000*, pages 519–528, Israel, March 2000.

[42] B. Fortz and M. Thorup. Optimizing OSPF/IS-IS weights in a changing world. *IEEE Journal on Selected Areas in Communications*, 20(4):756–767, May 2002.

[43] K.G. Ramakrishnan and M.A. Rodrigues. Optimal routing in shortest path data networks. *Lucent Bell Labs Technical Journal*, 6(1), 2001.

[44] A. Gunnar, H. Abrahamsson, and M. Söderqvist. Performance of Traffic Engineering in Operational IP-Networks: An Experimental Study. In *Proceedings of 5th IEEE International Workshop on IP Operations and Management*, Barcelona, Spain, October 2005.

[45] A. Sridharan, R. Guérin, and C. Diot. Achieving Near-Optimal Traffic Engineering Solutions for Current OSPF/IS-IS Networks. In *IEEE Infocom*, San Francisco, March 2003.

[46] David Applegate and Edith Cohen. Making Intra-Domain Routing Robust to Changing and Uncertain Traffic Demands: Understanding Fundamental Tradeoffs. In *Proceedings of ACM SIGCOMM*, Karlsruhe, Germany, August 2003.

[47] S. Balon, F. Skivee, and G. Leduc. How Well Do Traffic Engineering Objective Functions Meet TE Requirements? In *Proceedings of IFIP International Networking Conference*, Coimbra, Portugal, May 2006.

[48] C. Zhang, Z. Ge, J. Kurose, Y. Liu, and D. Townsley. Optimal Routing with Multiple Traffic Matrices: Tradeoffs between Average Case and Worst Case Performance. In *Proceedings of ICNP 2005*, Boston, USA, November 2005.

[49] C. Zhang, Y. Liu, W. Gong, J. Kurose, R. Moll, and D. Townsley. On Optimal Routing with Multiple Traffic Matrices . In *Proceedings of Infocom 2005*, Miami, USA, March 2005.

[50] H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, and A. Greenberg. Cope: Traffic engineering in dynamic networks. In *Proceedings of ACM SIGCOMM*, Pisa, Italy, September 2006.

[51] Anders Gunnar and Mikael Johansson. Robust routing under bgp reroutes. In *Proceedings of Globecom 2007*, Washington, DC, USA, 2007.

[52] Antonio Nucci, Supratik Bhattacharyya, Nina Taft, and Christophe Diot. IGP Link Weight Assignment for Operational Tier-1 Backbones. *IEEE/ACM Transactions on Networking*, 15(4), August 2007.

[53] Dahai Xu, Mung Chiang, and Jennifer Rexford. DEFT: Distributed exponentially-weighted flow splitting. In *IEEE Infocom*, Anchorage, Alaska, USA, May 6–12, 2007.

[54] Dahai Xu, Mung Chiang, and Jennifer Rexford. Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering. In *IEEE Infocom*, Phoenix, Arizona, USA, April 2008.

[55] J. Rexford. Network Protocols Designed for Optimizability. In *Proc. Conference on Information Sciences and Systems*, March 2006.

[56] Jing Fu, Peter Sjödin, and Gunnar Karlsson. Traffic engineering and routing in IP networks with centralized control. May 2008. IFIP Networking 2008.

[57] D. Mitra and K. G. Ramakrishnan. A Case Study of Multiservice, Multipriority Traffic Engineering Design for Data Networks. In *Proceedings of Globecom'99*, Brazil, 1999.

[58] I. Gojmerac, T. Ziegler, and P. Reichel. Adaptive Multipath Routing Based on Local Distribution of Link Load Information. In *Proceedings of QofIS 2003*, pages 122–131, Stockholm, Sweden, Oct 2003.

[59] J. Andres-Colas, F. J. Ramon-Salguero, A. Molins-Jimenez, and J. Enriquez-Gabeiras. Multipath Routing with Dynamic Variance. COST 279 Technical Report TD02043, 2002.

[60] M. A. Callejo-Rodriguez, J. Andres-Colas, G. Garcia de Blas, F. Javier Ramon-Salguero, and J. Enriquez-Gabeiras. A Decentralized Traffic Management Approach for Ambient Networks Environments. In *16th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2005)*, Barcelona, Spain, October 2005.

[61] S. Vutukury and J.J. Garcia-Luna-Aceves. A Simple Approximation to Minimum Delay Routing. In *Proceedings of ACM SIGCOMM'99*, Cambridge, Massachusetts, September 1999.

[62] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. Overview and principles of Internet traffic engineering. Internet RFC 3272, May 2002.

II

Included Papers

Chapter 6

Paper A: Using Empirical Distributions to Characterize Web Client Traffic and to Generate Synthetic Traffic

Henrik Abrahamsson and Bengt Ahlgren.
In *Proceedings of IEEE Globecom: Global Internet*, San Francisco, USA, November 2000.

Abstract

We model a web client using empirical probability distributions for user clicks and transferred data sizes. By using a heuristic threshold value to distinguish user clicks in a packet trace we get a simple method for analyzing large packet traces in order to get information about user OFF times and amount of data transferred due to a user click. We derive the empirical probability distributions from the analysis of the packet trace. The heuristic is not perfect, but we believe it is good enough to produce a useful web client model.

We use the empirical model to implement a web client traffic generator. The characteristics of the generated traffic is very close to the original packet trace, including self-similar properties.

6.1 Introduction

Measurements on the Internet backbone [1], [2] show that HTTP comprises approximately 70-75 % of the total traffic. To understand the behavior of the aggregated traffic, it is therefore important to understand how the HTTP traffic behaves. We have the goal of implementing a traffic generator which can be used to generate realistic best-effort background traffic in lab networks. When introducing multiple traffic classes, as in diffserv, the background best-effort traffic will to some extent disturb higher priority traffic, such as voice, depending on queue management and scheduling algorithms. To be able to make realistic lab experiments with traffic classes, we need a web traffic generator.

We start with a packet trace of web traffic. But we do not want to model the number, size and inter-arrival times of TCP packets since these quantities are governed by the TCP flow control and congestion control algorithms. The timing of a connection's packets as recorded in a trace reflects the conditions in the network at the time the connection occurred. Due to this adaptation to the network done by TCP, a trace of a connection's packets cannot easily be reused in another context, because the connection would not have behaved the same way in the new context [3]. Instead we use the packet trace to characterize the behavior at a higher level rather than at the packet level.

We base our web client model on user clicks and statistics of the amount of data transferred as a result of each click. In the packet trace we detect when a user clicks on a link to get the next web page and from that we deduce how much data that was transferred in response from the web server, as well as the time between the end of the transfer to the next click. This time of silence preceding a click is here called *user OFF time*. The packet trace analysis uses heuristics to deduce user clicks without the need to parse HTTP requests. This makes it possible to analyze very large traces and traces which only has the packet headers recorded.

We use the empirical model to implement a web client traffic generator. We show that the resulting aggregated traffic from many sources have the same properties, including self-similarity, as the traffic in the original trace.

The contributions of this paper include heuristics for detecting user clicks in a packet trace, a simple empirical web client model and a realistic web client traffic generator.

The remainder of the paper is organized as follows. Section 6.2 gives a brief introduction to the HTTP protocols, a description of the packet trace and the method used to extract information from the trace. The resulting empirical distributions are presented in Section 6.3 and synthetic traffic generation using

these distributions is described in Section 6.4. The paper is ended with related work and conclusions.

6.2 Analyzing web traffic

6.2.1 The HTTP protocols

The application-level protocol HTTP exists and is used in more than one version. There is yet no formal standard that everybody follows.

HTTP/1.0 [4] is a simple protocol. The web browser establishes a TCP connection to the web server, issues a request, and reads back the server's response. The server indicates the end of its response by closing the connection. When a browser using HTTP/1.0 fetches web pages it sets up a new TCP connection for each requested document. Web pages often have many embedded images, which each is retrieved via a separate HTTP request. Thus, to retrieve a web page with five images, six different TCP connections are required. The first TCP connection transfers an HTTP GET request to receive the HTML document that refers to the five images. A very simple browser would, when the HTML document is received, open one new TCP connection to get the first image. After sending the response the connection is closed by the server and another connection is opened to get the second image and so on. The use of a new TCP connection for each image serializes the display of the entire page. Netscape introduced the use of parallel TCP connections to compensate for this serialization. When the HTML document is received normally four TCP connections are opened in parallel for the first four images which decreases the transaction time for the user.

HTTP/1.1, as it is described in RFC 2616 [5], differs from HTTP/1.0 in numerous ways, both large and small. Of most interest here is the network connection management. The problem in HTTP/1.0 that a new TCP connection is required for each document is resolved by the use of persistent connections and the pipelining of requests on a persistent connection. Persistent connections means that the client and server keep a TCP connection open instead of the server closing the connection after sending the response. The same connection can be used to fetch several images and can be kept open even if the user clicks to another web page as long as the page is located on the same server. Pipelining means that a client can send an arbitrarily large number of requests over a TCP connection before receiving any of the responses. HTTP/1.0, in its documented form, made no provision for persistent connections but some

implementations use a Keep-Alive header to request that a connection persist.

6.2.2 The packet trace

To get information about user OFF times and the amount of data transferred due to a user click, web traffic was captured using *tcpdump* [6]. Figure 6.1 shows the network at SICS. The machine running *tcpdump* (called network

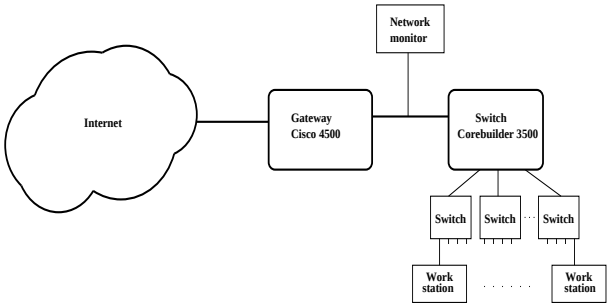


Figure 6.1: The network at SICS

monitor in the figure) was listening to the 100 Mb/s line connecting all workstations at SICS with the gateway. This was used to capture conversations between machines at SICS and the outside Internet world. Only traffic where users at SICS were clients was captured, not the HTTP traffic that arise from people outside visiting the SICS web pages. The packet trace was taken between 18:50:04 000222 and 11:17:51 000301 and includes 8317992 packets transferred between TCP port 80 on web servers and 181 different clients at SICS. The amount of HTTP traffic varies of course during the day and during the week (Fig. 6.2) depending on how many people are using the network. But also when the traffic is studied on lower time scales from hours down to milliseconds there is a lot of variation in the number of bytes and packets sent.

Figure 6.3 shows the traffic during one of the busiest hours where at most 52 client were active. A few years ago Leland *et al.* [7] showed that LAN traffic is bursty on many time scales in a way that can be well described using *self-similar processes* and later Crovella *et al.* [8] showed that this also holds for web traffic. The degree of self-similarity is expressed using the so called *Hurst* parameter. This parameter can take any value between 0.5 and 1 and the higher the value the higher the degree of self-similarity. For Poisson traffic the value is $H = 0.5$. An often used heuristic graphical method to estimate

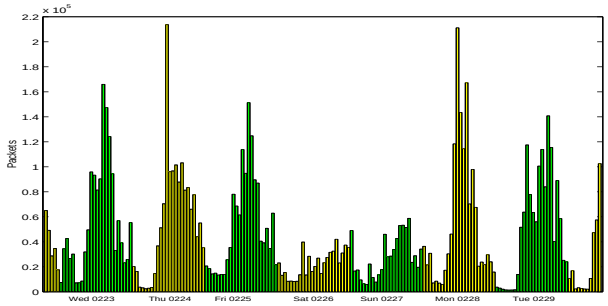


Figure 6.2: Packets per hour between 19:00 000222 and 11:00 000301

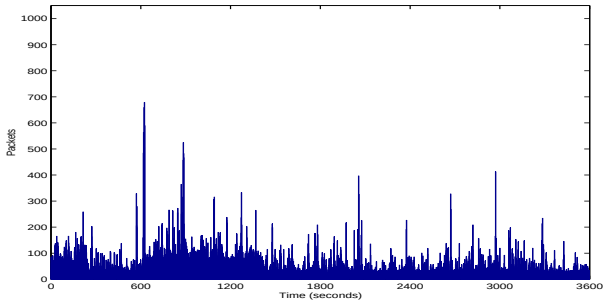


Figure 6.3: Packets per second 11:00-12:00 Mon 000228

the Hurst parameter is the Variance-Time plot which relies on the fact that a self-similar process has slowly decaying variances. For a detailed discussion of self-similarity and the methods used for estimating the Hurst parameter see Leland *et al.* [7] and Crovella and Bestavros [8]. Figure 6.4 shows an estimate of the Hurst parameter for the hour 11:00-12:00 000228 using the Variance-Time plot. The value is 0.87 so the traffic during that hour can be said to be self-similar meaning bursty on many time scales. Self-similarity expressed using the Hurst parameter seems to be a good way of describing the behavior of real web traffic and it would be good if the generated traffic also have the same properties.

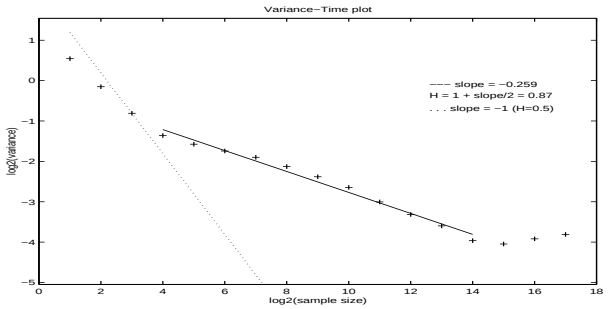


Figure 6.4: Estimate of the Hurst parameter

6.2.3 Detecting user clicks

When a user clicks on a link to get the next web page, the browser sends a HTTP request to the server. We want to detect these requests and separate them from requests for parts of a web page. To separate a request due to a user click from other requests the time of silence preceding it is investigated. A request is assumed to be due to a user click if it is preceded by enough time of silence, an interval here called T_{click}, where no HTTP traffic is sent to or from this client. The packet trace doesn't contain application level HTTP requests and responses, but only lower level TCP/IP packet headers. Different users use different browsers with different number of parallel TCP connections and where some use persistent connections and some don't. This means that the start and end of connections cannot be used to determine if a user have clicked on a link to fetch a new web page. Instead only the time between the last HTTP response (or request) and a new request is considered, irrespective of which TCP connection the client uses for the transfer. Since the HTTP client sends almost only requests, we assume that every TCP packet from a client - carrying some payload data (not pure acknowledgment or control packet) - is transferring a HTTP request. If the transfer of a TCP packet that carries a request is preceded by a period of T_{click} seconds where no data is transferred to or from this client then we assume that this packet represents a user click. The problem is to determine the value of T_{click}. The value should be large enough, so that requests for parts of the same web page is not counted as user clicks, and small enough to separate different user clicks.

Similar problems have been addressed by Mah [9] and by Crovella and Bestavros [8]. When investigating packet traces in order to determine the num-

ber of files per web page, Mah uses the threshold value 1 second to separate connections that belongs to different web pages. The main reason for the choice of this value was that users will generally take longer than one second to react to the display of a new page before they order a new document retrieval. When investigating causes of self-similarity in WWW traffic, Crovella analyses OFF times and concludes that times in the range of 1 ms to 1 second is likely to be strongly determined by machine processing and display time for data items that are retrieved, not due to users examining data.

For each request we calculated the time of silence preceding it. From these times the requests were sorted and counted. Figure 6.5 shows the result with

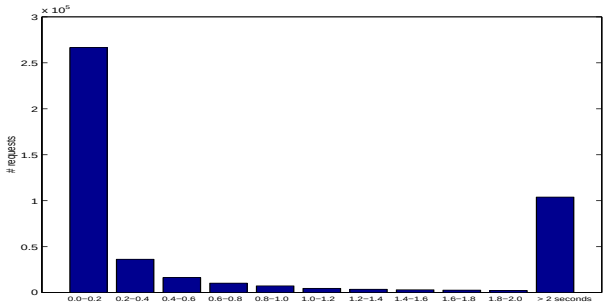


Figure 6.5: Time of silence preceding HTTP requests

time of silence in 0.2 second bins ranging from 0 to 2.0 seconds. We chose the value T_{click} = 1 second, even though the values in Figure 6.5 might suggest that an even smaller value would have been reasonable.

In order to validate that the method and threshold value described really gives reasonable results we used a proxy X-server that logged time-stamps on the mouse button-up events when using Netscape. This was used to log the actual clicks made and at the same time tcpdump was used to capture all web traffic to and from the client. The packet trace was analyzed using the threshold value T_{click} = 1 second in order to detect user clicks. The time-stamps of the detected clicks were compared to the time-stamps in the X-server log. The result is shown in Table 6.1. If a click detected in the trace has a time-stamp equal to (or very close to) a time-stamp in the log file it is called a *hit*. If a detected click in the trace does not correspond to a real click it is called a *false* click and if a click in the log file is not detected in the packet trace it is said to be *missed*.

# detected clicks in trace	629
# real clicks in log	532
Hits	519 (97%)
Missed	13 (3%)
False	110 (18%)

Table 6.1: Examination of the clicks detected

Approximately 97% of the real clicks were detected and 82% of all detected clicks were correct. It should be emphasized that since only quite a small number of clicks have been investigated and the timing of requests depends on the user, the machine used, what pages are visited and so on the results in Table 6.1 should only be seen as coarse estimates. A closer examination of the packet trace shows that twelve of the false clicks were due to retransmissions of requests but the main reason for the many false clicks is that requests for part of a web page is sometimes preceded by more than one second of silence and thus detected as user clicks. In general, clicks are missed because the client quickly clicks to navigate to another web page before the transfer of the previous one was completed. In that case there is no one-second interval of silence preceding the request so the click is not detected. So, not all but too many clicks are detected. A larger value of T_{click} would give less false but more missed clicks. The method used to detect clicks is not perfect but from the results in Table 6.1 it seems to be good enough to be useful.

6.2.4 User sessions

Since a client that begins with an hour of silence or takes a two week vacation is not very useful in a traffic generator we also need to break up the traffic into user sessions. The notion of a session is supposed to cover the time interval when a user is active and uses the browser to fetch and read web pages. This is vague and hard to define, especially in terms of packets sent and received. We define a session to be an interval in which a user creates WWW traffic without being silent for more than a certain time. That is, a session starts when the first web page is fetched (the first request is made) and ends when the last page is received (but not yet read). If no request or response is sent for a certain time, a threshold value here called $T_{session}$, then the next request is the start of a new session. We used the value $T_{session}$ = 15 minutes.

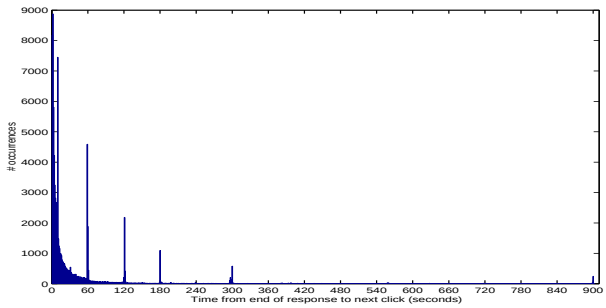


Figure 6.6: Histogram of time between end of response and next click

6.2.5 Data analysis

Awk was used to extract the needed information from the tcpdump file. The extracted data was later investigated further using Matlab. Only information about packets carrying payload data was extracted. The input to Matlab was a matrix where each such packet was represented with a time-stamp in microseconds, a unique client id, direction (client request or server response), and the packet size without headers. A Matlab program was written which for each client went through the times between requests and responses and used T_{click} to detect user clicks and determine user OFF-times and the amount of data transferred as response to a user click.

6.3 Empirical distributions

In this section, we use the packet trace and the heuristics from the previous section to develop the two empirical distributions needed to model web client traffic.

6.3.1 OFF times

In Figures 6.6 and 6.7 a histogram and the cumulative distribution function (CDF) of the time from the end of the response to the next user click are shown. There were a total of 90621 user OFF-times in the data set. The minimum time was 1.000003 seconds, just above the T_{click} threshold of one second. The maximum is determined by the value of $T_{session}$=15 minutes. The median

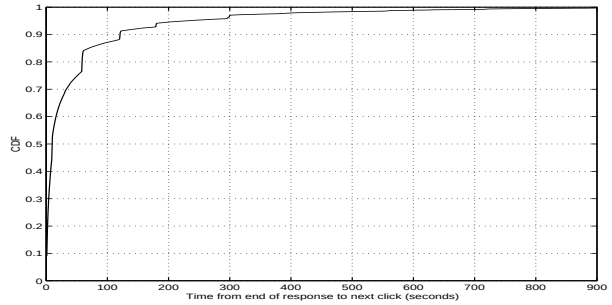


Figure 6.7: Empirical CDF of time between end of response and next click

time was 9.8 seconds and the mean 49.39 seconds with a standard deviation of 109.7 seconds. The coefficient of variation was 2.2.

There are several peaks in the histogram, most noticeable at 10, 60, 120, 180 and 300 seconds. A closer examination of the packet trace shows that for each of these peaks there is a single client that causes most of them. For instance, the peak at 10 seconds originates from a client that for 12 hours repeatedly sends requests to check if the web page has been modified and the peak at 300 is due to somebody updating their stock-exchange rates every five minutes. It is not obvious whether these periodic OFF-times should be included or characterized as anomalies and thus be removed from the data set. The requests are not really user clicks but the OFF times are a part of the real traffic so we let them contribute to the empirical distribution in Figure 6.7.

6.3.2 Amount of data transferred due to a single user click

The amount of data transferred from servers as response to a single user click varies a lot. On one occasion 31940163 bytes were transferred and at other times no data at all was received by the client. In Figure 6.8 only the part of the CDF that covers values below 250000 bytes is shown. The median was 7145 bytes and the mean was 39142 bytes with a standard deviation of 317753 bytes. The coefficient of variation was 8.1.

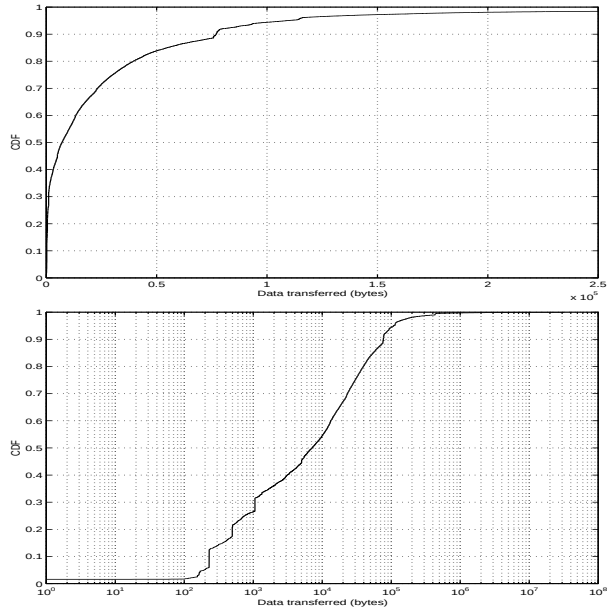


Figure 6.8: Empirical CDF of the amount of data transferred as response to a user click (linear x-axis at top and logarithmic at bottom).

6.4 Generating traffic

By using the distributions in Section 6.3, traffic can be generated that resembles a number of users surfing the web — reading web pages (OFF times) and clicking on links to get the next one (data transfer).

6.4.1 The traffic generator

The traffic generator was implemented in the C programming language and has a client and a server part. Values from the empirical distribution for OFF-times (Fig. 6.7) and data transferred (Fig. 6.8) was pre-computed and written to a file using the inverse transformation method described, for instance, by Jain [10]. The client reads from the file the OFF time and the amount of data that should be transferred and sends the latter as a request to the server. The server side just accepts requests and replies by sending the demanded amount of data. The

number of different clients is given as input to the program and each client is represented by a process that repeatedly goes through the loop of requesting and receiving data according to the distribution of data transferred due to a single user click and then goes to sleep for a time described by the distribution of OFF times until the next request is made.

6.4.2 Evaluation

Traffic resembling 60 clients was generated on a 10 Mb/s link between two machines. In order to validate that the OFF times and the amount of data transferred follows the distributions the generated traffic was captured using tcpdump and analyzed in the same way as the original packet trace. Figures 6.9 and 6.10 show the distributions for the generated traffic and a comparison with

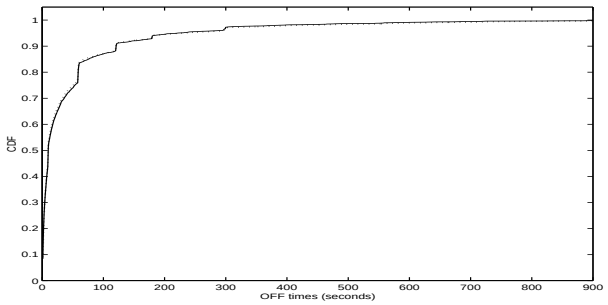


Figure 6.9: OFF times

the originals (dotted line). In the traffic generator no request is sent for a zero bytes response so the CDF for the amount of data transferred lies somewhat lower than the original.

There is a lot of variation in the number of packets that are transferred in one second (Fig. 6.11) and a Hurst parameter value of 0.76 (Fig. 6.12) indicates that the generated traffic, like real web traffic, is bursty on many time-scales.

6.5 Related work

Two approaches can be used to generate network traffic that imitates real web traffic. The first is simply to replay packet traces of real web traffic. But, because of TCP's flow and congestion control the timing of packets in a trace

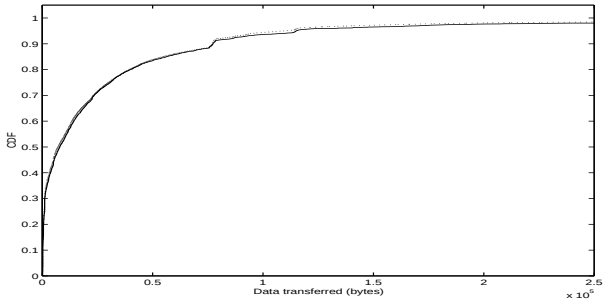


Figure 6.10: Data transfered

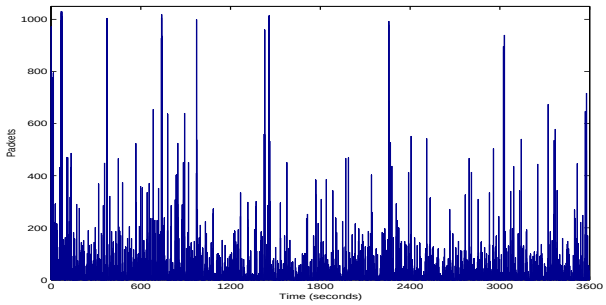


Figure 6.11: Packets per second

reflects the condition in the network when the trace was taken and this timing would not be the same in another context. The alternative is to gather information about, and mathematically describe, those aspects of the web traffic that one believe is most important and from this model generate traffic. This approach was used by Barford and Crovella [11].

To get information about web traffic three different approaches have been widely used: server logs, client logs and packet traces. Server logs cannot easily be used to describe the client side since a client usually accesses many different web servers. To capture the client accesses between multiple servers, client logs can be used. This approach requires that browsers can log their requests, that the source code for the browser is available so that logging can be added, or that some other way to log the clients behavior is available. Catledge and Pitkow [12], Cunha *et al.* [13] and Crovella and Bestavros [8] use instru-

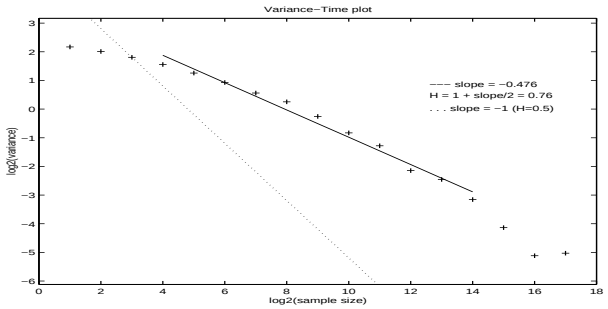


Figure 6.12: Estimate of the Hurst parameter

mented versions of the Mosaic web browser. Barford *et al.* [14] use HTTP proxies to track all documents referenced by unmodified Netscape Navigator clients. The third approach of gathering data, and the method used here, is to analyze packet traces taken from a subnet carrying HTTP traffic. This method was used by Stevens [15] to analyze the traffic arriving at a server, and by Mah [9] to model the client side of the HTTP traffic. A further step is taken by Anja Feldman [16] when extracting full HTTP level as well as TCP level traces via packet monitoring.

6.6 Conclusions and future work

We have presented an empirical model for web client traffic. The model is based on user click behavior combined with statistics of the amount of data transferred per click. The user clicks, or actually the silence times before a click, and the amount of data are modeled using cumulative distribution functions. By using a heuristic threshold value to distinguish user clicks in a packet trace, we get a simple method for analyzing large packet traces without the need for parsing HTTP requests. The result of the analysis are data defining the two distribution functions. The simplicity of the heuristic packet trace analysis may have a price in accuracy. A verification, however, shows that the heuristics correctly detect 82 % of the actual user clicks from the packet trace. We believe that this is sufficiently accurate to produce a good empirical model.

We have implemented a web client traffic generator which takes the cumulative distribution functions as input. We have shown that the generated synthetic traffic have the same characteristics as the original packet trace by

applying the same analysis to a trace of the generated traffic. We have also verified that the aggregated generated traffic from many clients has self-similar properties just like the original trace.

Future work include analyzing packet traces from more networks and comparing the resulting distribution functions. We plan to use the traffic generator to generate best-effort background traffic in lab experiments with voice-over-IP and multiple traffic classes. We also plan to release the source code to the analysis software and the traffic generator shortly.

Acknowledgments

The authors would like to thank Assar Westerlund for providing the logging proxy X-server and Peter Ehlin and Björn Grönvall for their help when taking the packet traces.

Bibliography

[1] K. Claffy, G. Miller, and K. Thompson. The nature of the beast: recent traffic measurements from an internet backbone. http://www.caida.org/Papers/Inet98.

[2] G. J. Miller K. Thompson and R. Wilder. Wide-area traffic patterns and characteristics (extended version). *IEEE Network*, 11(6):10–23, 1997.

[3] V. Paxson and S. Floyd. Why we don't know how to simulate the Internet. In *Proceedings of the 1997 Winter Simulation Conference*, Atlanta, GA, USA, 1997.

[4] RFC 1945 *Hypertext Transfer Protocol – HTTP/1.0*. Available at: http://www.ietf.org/rfc/rfc3272.txt.

[5] RFC 2616 *Hypertext Transfer Protocol – HTTP/1.1*. Available at: http://www.ietf.org/rfc/rfc2616.txt.

[6] V. Jacobson, C. Leres, and S. McCanne. tcpdump software. This software is available at ftp://ftp.ee.lbl.gov/tcpdump.tar.Z.

[7] W. Leland, M. Taqqu, W. Willinger, and D. Wilson. On the self-similar nature of ethernet traffic (extended version). *ACM Transactions on Networking*, 2(1), 1994.

[8] M. E. Crovella and A. Bestavros. Self-similarity in World Wide Web traffic: Evidence and possible causes. *IEEE/ACM Transactions on Networking*, 5(6):835–846, 1997.

[9] B. A. Mah. An empirical model of HTTP network traffic. In *Proceedings of INFOCOM '97*, Kobe, Japan, April 1997.

[10] R. Jain. *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, New York, 1991.

[11] P. Barford and M. Crovella. Generating representative Web workloads for network and server performance evaluation. In *Proceedings of Performance '98/ACM SIGMETRICS '98*, pages 151–160, Madison, Wisconsin, USA, 1998.

[12] L. D. Catledge and J. E. Pitkow. Characterizing browsing strategies in the World Wide Web. In *Proceedings of the third International World Wide Web Conference on technology, tools and applications*, Darmstadt, Germany, April 1995.

[13] C. R. Cunha, A. Bestavros, and M. E. Crovella. Characteristics of WWW client-based traces. Technical Report Technical Report BU-CS-95-010, Computer Science Department, Boston University, July 1995.

[14] P. Barford, A. Bestavros, A. Bradley, and M. Crovella. Changes in Web client access patterns: Characteristics and caching implications. *World Wide Web*, 2:15–28, 1999. Special issue on Characterization and Performance Evaluation.

[15] W. R. Stevens. *TCP/IP illustrated, Vol. 3, TCP for Transactions, HTTP, NNTP, and the UNIX Domain Protocols*. Addison-Wesley, 1996.

[16] A. Feldmann. BLT: Bi-Layer Tracing of HTTP and TCP/IP. In *Proceedings of 9th International World Wide Web Conference*, Amsterdam, May 2000.

Chapter 7

Paper B: TCP over High Speed Variable Capacity Links: A Simulation Study for Bandwidth Allocation

Henrik Abrahamsson, Olof Hagsand, and Ian Marsh.
In *Protocols for High Speed Networks Workshop*, Berlin, Germany, April 2002.

Abstract

New optical network technologies provide opportunities for fast, controllable bandwidth management. These technologies can now explicitly provide resources to data paths, creating demand driven bandwidth reservation across networks where an applications bandwidth needs can be meet almost *exactly*. Dynamic synchronous Transfer Mode (DTM) is a gigabit network technology that provides channels with dynamically adjustable capacity. TCP is a reliable end-to-end transport protocol that adapts its rate to the available capacity. Both TCP and the DTM bandwidth can react to changes in the network load, creating a complex system with inter-dependent feedback mechanisms. The contribution of this work is an assessment of a bandwidth allocation scheme for TCP flows on variable capacity technologies. We have created a simulation environment using ns-2 and our results indicate that the allocation of bandwidth maximises TCP throughput for most flows, thus saving valuable capacity when compared to a scheme such as link over-provisioning. We highlight one situation where the allocation scheme might have some deficiencies against the static reservation of resources, and describe its causes. This type of situation warrants further investigation to understand how the algorithm can be modified to achieve performance similar to that of the fixed bandwidth case.

7.1 Introduction

Reliable transfer of data across the Internet has become an important need. TCP [1] is the predominant protocol for data transfer on the Internet as it offers a reliable end-to-end byte stream transport service. Emerging optical networking technologies provide fast, cheap and variable capacity bandwidth links to be setup in milliseconds allowing data-driven virtual circuits to be created when needed. One example of an application that could use such a service is the backup of critical data.

Exact allocation of bandwidth to TCP flows would alleviate complex traffic engineering problems such as provisioning and dimensioning. Allocating bandwidth to TCP is a complex problem; the TCP congestion control mechanism plus network dynamics can make exact allocation for TCP data flows difficult. The contribution of this paper is the performance evaluation of an estimation algorithm, which measures the rate of TCP flows and allocates capacity on a DTM network.

Dynamic Synchronous Transfer Mode [2] [3] is a gigabit ring based networking technology that can dynamically adjust its bandwidth. DTM offers a channel abstraction, where a channel consists of a number of slots. The number of slots allocated to a channel determines its bandwidth. The slots can be allocated statically by pre-configured parameters, or dynamically adjusted to the needs of an application. In DTM it is possible to allocate a channel to a specific TCP connection, or to multiplex several TCP connections over the same channel. We mostly investigated cases where each TCP connection is assigned to a separate channel, but show one case in which two TCP connections compete for a single channel. The DTM link capacity is only allocated in the forward direction in this study, we have not performed any allocation for TCP acknowledgements.

TCP uses an end-to-end congestion control mechanism to find the optimal bandwidth at which to send data. In order to get good throughput with TCP operating over a technology such as DTM, it is important to understand the dynamic behaviour of the two schemes, especially when evaluating a bandwidth allocation strategy. TCP is capable of adjusting its *rate* whilst DTM is capable of changing its *capacity*. In dynamically interacting systems, it is possible to create unwanted oscillations resulting in under allocation or over allocation of bandwidth to TCP flows. In order to evaluate the performance of the DTM bandwidth allocator, we have implemented the algorithm in the network simulator ns-2. We have performed a number of simulations that include single and multiple TCP flows, links with varying delay characteristics, different buffer

sizes, plus TCP Reno and Tahoe variants.

Section 7.2 outlines DTM and our estimation algorithm, simulation experiments are given in Section 7.3, related work follows in Section 7.4, and finally conclusions and a discussion are given in Section 7.5.

7.2 Dynamic Synchronous Transfer Mode

DTM uses a TDM scheme where time slots are divided into control and data slots. The control slots are statically allocated to a node and are used for signalling. Every node has at least one control slot allocated that corresponds to 512 kbps of signalling capacity. The data slots are used for data transmission and each slot is always owned by a node. A node is only allowed to send in slots that it owns. The ownership of the slots is controlled by a distributed algorithm, where the nodes can request slots from other nodes. The algorithms for slot distribution between the nodes affect the network performance. Each slot contains 64 bits and the slots are grouped in 125 microsecond long cycles. The bit rate is determined by the number of slots in a cycle, so one slot corresponds to a bit rate of 512 kbps. By allocating a different numbers of slots, the transmission rate for a channel can be changed in steps of 512 kbps.

7.2.1 TCP Rate Estimation and DTM Capacity Allocation

TCP's rate is simply estimated as the number of *incoming* bytes per second. The algorithm which is presented next calculates the rate by dividing the number of bytes by the time elapsed. The rate of each flow is calculated ten times per second, i.e. every 100 ms. This value has been chosen as a compromise between good measurement granularity and processing overhead. DTM technology however, has the ability to sample flows up to gigabit speeds, i.e. at sampling rates higher than 100 ms. Actual slot allocation or changes are done only *once* every second, this is slightly coarser due to the overhead of nodes potentially having to negotiate slots.

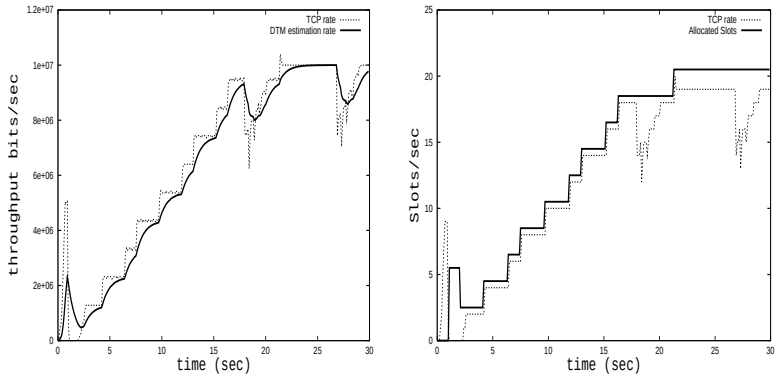
We now describe the TCP bandwidth estimator. Figure 7.1 shows the algorithm used to estimate the rate of a given flow. As stated, every 100 ms the estimator measures the rate `new` in bits per second and compares it with the previous value, `current`. A delta of the difference is reduced by `DTM_SHIFT` in the algorithm. Note this delta is simply shifted, keeping the complexity of the calculation to a minimum. In this case it is three, so the current value is changed by one eighth towards the recently measured flow value, as shown

```
dtm_calc_bw ( new ) {
DTM_SHIFT = 3
MARGIN = 0.75
CORRIDOR = 2

/* first half - Move last estimate closer */
diff = new - current
if ( diff < 0 ) {
   diff = (-diff) >> DTM_SHIFT
   current = current - diff // Decreasing
} else {
   diff = diff >> DTM_SHIFT
   current = current + diff // Increasing
}
curr_slot = current / slot_bw

/* Second half - Last estimate within bounds ? */
if ( curr_slot > upper_bound ) || (curr_slot < lower_bound ) {
   dynBw = curr_slot + MARGIN + ( CORRIDOR / 2 )
   /* only change bw once per sec */
   change_link_bw (dynBw)
   }
}
```

Figure 7.1: Algorithm for bandwidth estimation

in the first half of the algorithm. This shift effectively determines how aggressively TCP's rate can be tracked. This default value has been chosen experimentally, as DTM is a deployed technology. The technical report version of this paper shows the affect of using other values [4]. Finally the units are changed from bits per second to slots per second by dividing the rate by the channel bandwidth and assigning this value to the variable `curr_slot`.

The second half of the algorithm determines whether it is necessary to change the slot allocations. The current slot value is compared to upper and lower bounds before making any changes. An offset, 0.75 of a slot, `MARGIN` equivalent to 394 kbits, is added to the TCP throughput estimate so the DTM allocation will be a little over the estimated rate. Figure 7.2 shows two plots us-



(a) Estimating the TCP Rate (b) DTM slot allocation

Figure 7.2: Measured Throughput and Slot Allocation

ing the topology shown in Figure 7.3, the leftmost plot is the actual measured bandwidth of a single TCP flow. The right plot shows the effect of adding `MARGIN` and measuring the rate in slots. If the allocation was based purely on this estimate it would under allocate bandwidth, causing TCP reduce its window because of congestion on the link. The rightmost graph is coarser due to the second granularity of the bandwidth changes. The plots illustrate how the estimation can be used to give TCP the bandwidth it needs and hence maximise throughput. One can also see in this figure that estimation starts after 100 ms but a change is not applied to the offered bandwidth before the first second. Note also the y-axis in Figure 7.2b) is in slots per second and not bits per second as in the left figure. Additionally, a `CORRIDOR` is an amount the estimate is allowed to vary before slots are added or decreased for a channel. This is not visible in the plots but will be illustrated later. The purpose is to avoid small fluctuations causing unnecessary costly slot allocation changes. As mentioned, slot changes can be time consuming due to the distributed nature of DTM [5].

7.3 Simulation Tests

This section presents simulation results that show how the DTM estimation algorithm adapts the offered bandwidth to TCP flows. Figure 7.3 shows the topology we used for the following simulations. The 5 Mbits per second link between nodes two and three is the bottleneck link. The link between nodes

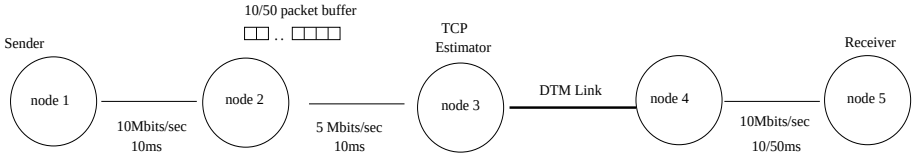
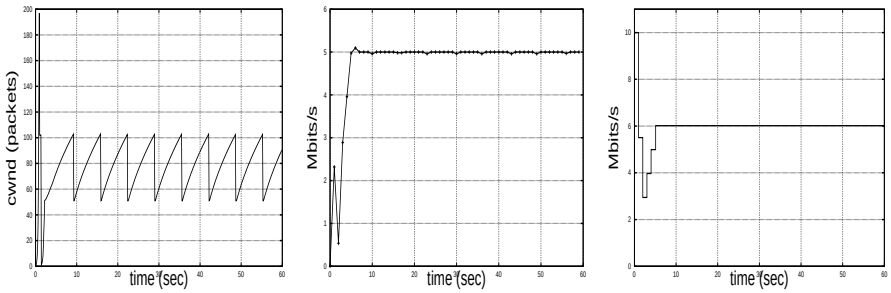


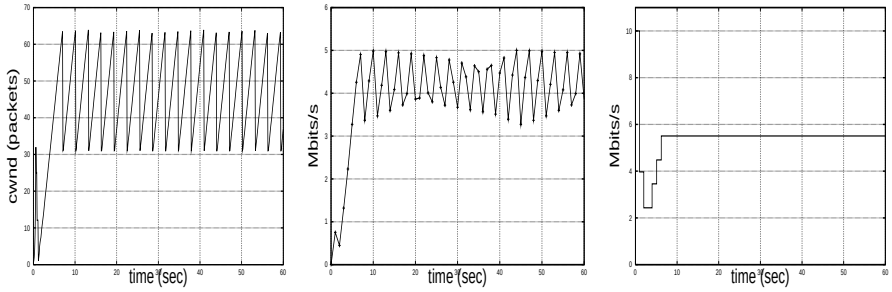
Figure 7.3: Simulation topology



(a) Congestion Window (b) Throughput (c) DTM allocation

Figure 7.4: Dynamically allocated bandwidth on a DTM link (50 packet queue)

three and four is the DTM link with dynamically allocated bandwidth. Initially the DTM link is set to 10 Mbits per second. This value was chosen simply for convenience, since simulating a 622 Mbits per second link with large bandwidth flows is not feasible in a packet level simulator like ns-2. The other two links also have a capacity of 10 Mbits per second. A bulk transfer TCP Reno flow was setup between nodes one and five and the throughput measured at node three, in order to allocate bandwidth on the outgoing DTM link. In this first simulation the queue length in node 2 was set to 50 packets, figure 7.4 shows the result. In congestion avoidance the TCP flow increases the congestion window by the maximum segment size bytes each RTT seconds. However, the increase is not made each RTT. Instead TCP will increase $MSS/congestion$ window bytes each time an ACK is received. This means that after RTT seconds, the congestion window was increased by MSS bytes. This continues until the TCP flow has filled the buffer space at the bottleneck link, resulting in a packet drop. TCP Reno, using fast retransmit and fast recovery, then reduces the congestion window by half and continues with conges-

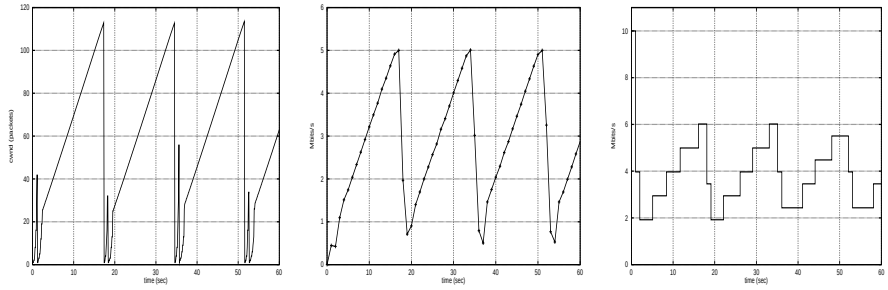


(a) Congestion Window (b) Throughput (c) DTM allocation

Figure 7.5: Dynamically allocated bandwidth on a DTM link (10 packet queue)

tion avoidance. The congestion window, therefore, follows a sawtooth curve. If enough buffer space is available at the bottleneck link, the rate of the TCP flow, perceived after the second link, is not affected when the congestion window is reduced. This mechanism and result can be seen in left and middle plots of Figure 7.4. The rightmost plot shows the dynamically allocated bandwidth on the DTM link. It can be seen that TCP actually manages to get about one Megabit per second more on the DTM link due to the extra capacity allocated to the flow through the addition of `MARGIN`. It should be stated in a real deployment of TCP over DTM that this value is settable by network operators. Its affect can be tested in simulation environments such as this if necessary.

Figure 7.5 shows the results when the queue size at the bottleneck link is limited to ten packets. This could be the case if a static allocation over the DTM network has been setup. Now the rate of the TCP flow changes with the congestion window, but the changes are too small to affect the dynamic allocation of bandwidth. This is due to the corridor mentioned earlier to avoid small changes from incurring changes in the slot allocation scheme. Figure 7.6 shows the case in which the simulation with a small queue size and a 50 ms link delay has been repeated using TCP Tahoe instead of TCP Reno. TCP Tahoe only relies on the retransmission timer and does not use fast retransmit. When a packet is dropped, the congestion window is set to one and slow-start is invoked. We can see that the allocation on the DTM link closely follows the sharp saw tooth behaviour of TCP Tahoe Figure 7.6c).



(a) Congestion Window (b) Throughput (c) DTM allocation

Figure 7.6: Tahoe TCP on a DTM link (10 packet queue)

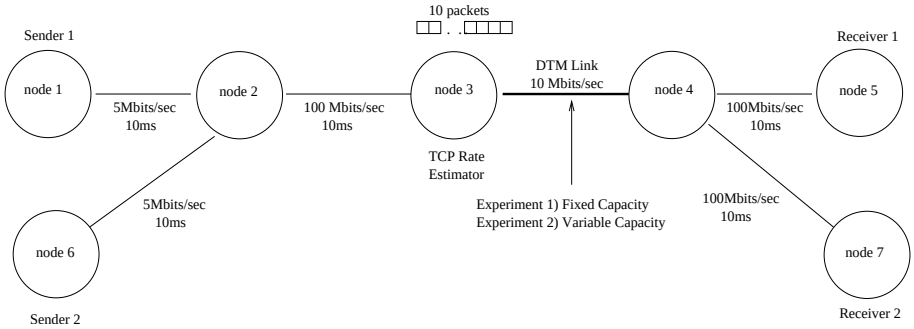


Figure 7.7: Simulation topology with two flows

7.3.1 Two Flows Per Channel and Small Router Buffers

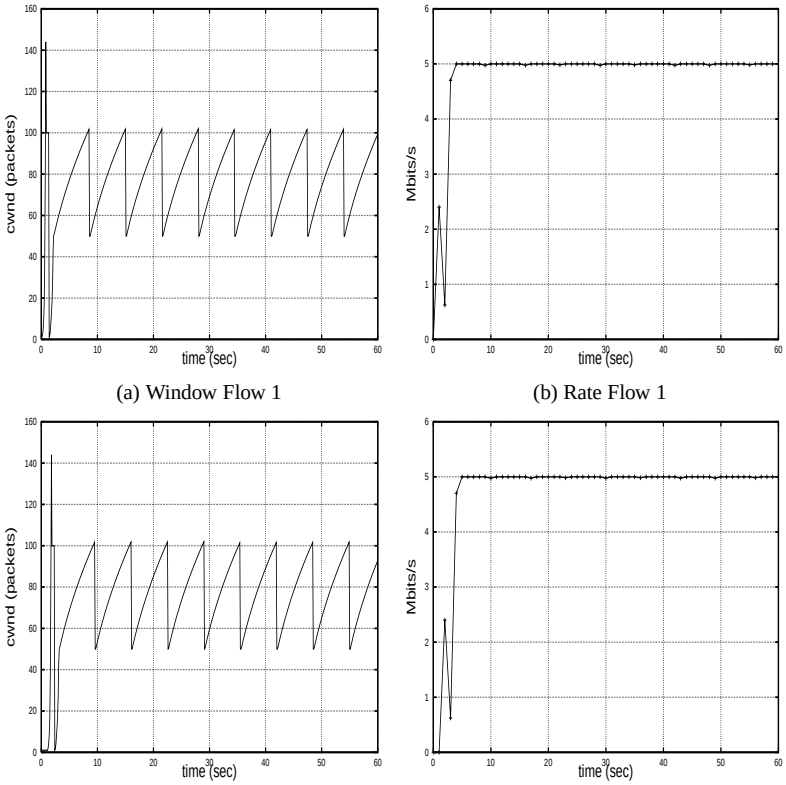
So far, we have shown cases where the dynamic allocation of bandwidth has allowed TCP to maximise its throughput. We illustrate one case next when the algorithm has weaknesses to allocate sufficient bandwidth to two TCP Reno flows. In this scenario the fixed link case performs better. Figure 7.7 shows the topology that we used. It differs from previous simulations in that the flows have their own input buffer at node two but share a common output buffer in the same node. This buffer is also served ten times faster than in previous cases by the fact that the link feeding the DTM network was set to 100 Mbits per second. In this case the queue length of a DTM link, node three, was limited

to ten packets.

Figure 7.8 shows the results when the link capacity between nodes 3 and 4 was fixed at 10 Mbits per second. We can see that both flows manage to reach their 5 Mbits throughput, effectively sharing equally one DTM channel. If we now turn our attention to the same simulation but replace the static link between nodes three and four with a variable one the results are quite different. Figure 7.9 shows the dynamically allocated bandwidth on the DTM link. Neither of the flows manage to reach 5 Mbits per second on their output links. In this case packets are being dropped in the output buffer of node three. This can be seen in the congestion windows of the two flows, they never manage to maintain the size of the static case, about 100 segments. The problem in this case is the estimation algorithm should not *decrease* the estimation *if* packets are being dropped. The algorithm is symmetric, it increases or decreases depending on the measured rate. Additionally the effect of the short queue does not help, there is not sufficient pressure with a small queue to keep the rate up, with a larger buffer there is more pressure due to accumulated packets. Interestingly, the algorithm actually correctly allocates for the observed throughput, however does not maximise the TCP throughput.

Some researchers have put forward TCP variants which are capable of estimating the bandwidth such as TCP Westwood [6], which do not solely rely on packet loss for congestion. It is not clear whether TCP variants such as this will improve on the situation above without substantial simulations. However other TCP variants would be worthy of investigation. The important point to note is that it is important to detect loss early and this can be done by monitoring queues for the local node or even via mechanisms such as RED or ECN for upstream nodes.

Our conclusion is that in the face of loss at a node the estimation algorithm should not decrease to allow TCP to recover. So a special case for loss could be introduced, *during loss* the rate could be estimated but no action is taken to adjust the bandwidth. One other solution would be to adjust the rate more slowly when allocating *less* bandwidth. This is trivial in the present scheme, the `DTM_SHIFT` variable can be split into `DTM_SHIFT_INC` and `DTM_SHIFT_DEC` where a decrease in bandwidth takes place at a slower rate. This might have adverse affects on a normal behaving system so, once again would need to be validated with further simulations. We note that [7] also performed comparisons with the fixed link case and show in one experiment that the dynamically allocated link was not able to achieve the throughput of a fixed link case. With a certain selection of parameters it was only possible to allocate slightly over half of the fixed link case. We conclude that it is not



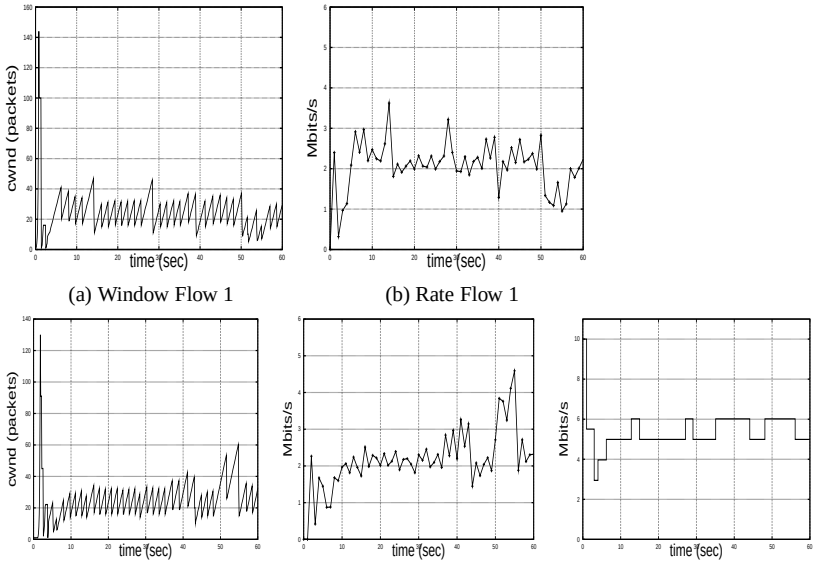
(a) Window Flow 1

(b) Rate Flow 1

(c) Window Flow 2

(d) Rate Flow 2

Figure 7.8: Experiment 1 static link: The senders do not drop packets at the ingress node and achieve their constrained link throughput 5Mbits per second.



(a) Window Flow 1 (b) Rate Flow 1

(c) Window Flow 2 (d) Rate Flow 2 (e) Aggregated DTM rate

Figure 7.9: Experiment 2 Dynamic Link: Allocated bandwidth on DTM link. Congestion window and measured rate for two TCP Reno flows with limited queue size at the DTM link.

always possible to perform as the fixed link case, however comparisons, wherever possible should be performed. The savings of allocating just the required bandwidth, however can be considerable. This was the methodology employed during our investigation.

7.4 Related Work

Work on estimating and maximising TCP throughput for variable capacity links is relatively scarce. However comprehensive studies have been done related to the performance of TCP on ATM networks [8] [9] [10]. The main conclusions of the works are similar, the traffic classes of ATM are poorly suited to the bursty needs of TCP, due to the traffic contracts needed by ATM classes. The conclusion of [8] is that the complexity of choosing traffic parameters for ABR is not in proportion to the benefits of carrying TCP/IP traffic. The CBR class is too simple for TCP, as only the peak rate is specified. Most of the DTM research in this area focuses on the distributed slot allocation for example [5].

Clark and Fang propose a framework for allocating bandwidth to different users during congestion [11]. The focus of the work is TCP bulk-data transfers. The authors attempt to keep TCP flows in congestion avoidance in the best case, and fast recovery phase in the worst case, by avoiding dropping several packets of the same flow in the same RTT. The conclusions of the given work are similar to those of [8], that TCP connections can have difficulties to fill their alloted bandwidth. The work resembles ours in that they attempt to allocate bandwidth between different flows in a fair manner. It differs from ours in that we assume that the network can change its offered bandwidth and we focus on maximising TCP throughput, rather than trying to maintain a TCP state in the face of adverse network conditions. In addition, we allocate bandwidth to flows not only when the network is congested but also in normal situations as well.

Sterbenz and Krishnan investigate TCP over Load-Reactive Links in a ICNP publication [12] and a technical memorandum [7]. They use a hysteresis control mechanism for capacity allocation. Buffer levels are monitored (as in [13]) and if the occupancy is greater than a threshold the capacity is increased and vice versa. This approach is not the same as ours, we measure the rate of incoming TCP flows at the router before the DTM link rather than the buffer level in the router at the outgoing DTM link. Their scheme is dependent on keeping buffers occupied all the time, otherwise the link capacity will fall and hence the throughput. A single TCP flow is simulated and the authors state that

the control parameters should be carefully chosen. Poor parameter choice can have the opposite effect, resulting in TCP not being able to operate, as stated previously. The work resembles ours in that a method is presented to react to network load and allocate bandwidth for TCP accordingly. It differs in that we measure the throughput of individual flows and allocate bandwidth from this measurements, where they use the buffer length as a measure of the load. Our system is less scalable, but more accurate, as we can ascertain exactly the bandwidth of the incoming TCP connections. Also there is no need to keep buffers occupied to allocate bandwidth for TCP connections. Also there is no potential interaction between several dynamic allocation schemes running on the same node.

Lundqvist evaluates different algorithms for bandwidth allocation for DTM channels transporting IP traffic [13]. The algorithms were assessed with respect to throughput, delay and bandwidth changes per second. TCP rate adjustment is done by placing the incoming packets into a buffer and adding and removing slots if the level of the buffer exceeds continuously maintained threshold values. He concludes that adaptive strategies are recommended for TCP, however too frequent changes can be undesirable in a DTM network due to the processing cost. The main conclusion from this work is that the choice of algorithm can play a significant role in the performance. This work is similar to ours in that the goal is a slot allocation for TCP traffic over DTM. We also agree it is important to keep the computational complexity low and DTM bandwidth changes as infrequent as possible. It differs from ours in that we measure the rate of each TCP flow, whilst he looks at the outgoing buffer length as a sign to increase or decrease the number of slots. We look more into network scenarios such as different link delays, buffer lengths and use two different TCP types, TCP Reno and Tahoe.

7.5 Conclusions

We have analysed a complex problem, allocating bandwidth to a protocol that can adapt its rate. The benefits of guaranteeing throughput for an application using TCP can be very beneficial, in particular the cost savings when paying per unit of transmission. The goal was to investigate the behaviour of our bandwidth estimation scheme, its affect on TCP *and* on a network that can vary its capacity, in this case DTM. Our work however is not only limited to DTM technology, we can draw the same conclusions about TCP performance on any high speed network technology that offers variable capacity.

We have written a simulation environment using ns-2, and found that in almost all cases, TCP could be allocated a share of the channel identical to its measured throughput on a fixed network. We identified one scenario in which the algorithm could be improved, when packets are dropped at a router with a small buffer. In this situation the estimation algorithm should not reduce the offered bandwidth further, resulting in less offered bandwidth and further packet loss. Instead it should allow TCP to find the new capacity available in the network. The combination of the small buffer size plus high speed input link aggravates this observed deficiency.

There are some other open issues, the system as described relies on measuring individual TCP flows, therefore methods that encapsulation such as MPLS would hinder us from measuring single flows. An alternative is to monitor and measure aggregate flows, however this was not the focus of our work. DTM is a fast MAN technology and can monitor flows at gigabit speeds, however if a bandwidth allocation scheme would be used in a non-DTM environment, especially in a backbone, some consideration would be needed for the sampling and measuring rate one can achieve with thousands of TCP flows. Aggregation of flows in this context would be a viable alternative.

We have only considered bulk data transfers, as the scheme measures flow bandwidths, it is not feasible to allocate bandwidth for all TCP flows, particularly http transfers as most data is transferred during the slow-start phase of a TCP connection, e.g. banners, buttons etc. Another issue is capacity provisioning for ACKs, we assumed the return path for acknowledgements is not constrained. In our experiments we had a return channel of 512 kbits per second which was more than adequate to support the forward data rates we were using, a maximum of 100Mbits per second. Further investigation is needed to state where problems could arise as well as potential solutions.

We have not considered well known scenarios such as satellite links with large bandwidth-delay products or more interestingly, where the control loops are sensitive to delay. We believe some benefit would be gained by looking at this problem (and others with time sensitive mechanisms) from a control theory perspective rather than the traditional networking approach, Westwood referenced earlier, takes exactly this approach.

In a simulation environment the parameter space is large. Due to space limitations we have only discussed a key subset of possible buffer sizes, link bandwidths, link delays and TCP variants. Parameters that are worthy of further investigation include sampling times and estimation thresholds. Further results, plus validation tests for using ns-2 in these kind of simulations, can be found in the technical report [4].

7.6 Acknowledgements

We would sincerely like to acknowledge the Computer and Network Architecture lab at SICS, and the Laboratory of Communication Networks at KTH, Royal Institute of Technology, Stockholm.

Bibliography

[1] J. Postel. Transmission control protocol. Request for Comments 793, Internet Engineering Task Force, September 1981.

[2] L. Gauffin, L. H. Hakansson, and B. Pehrson. Multi-gigabit networking based on DTM. 24(2):119–130, April 1992.

[3] Christer Bohm, Markus Hidell, Per Lindgren, Lars Ramfelt, and Peter Sjödin. Fast circuit switching for the next generation of high performance networks. 14(2):298–305, February 1996.

[4] Henrik Abrahamsson and Ian Marsh. Dtmsim - dtm channel simulation in ns. Technical Report T2001:10, SICS – Swedish Institute of Computer Science, November 2001.

[5] Csaba Antal, József Molnár, Sándor Molnár, and Gabor Szabó. Performance study of distributed channel allocation techniques for a fast circuit switched network. *Computer Communications*, 21(17):1597–1609, November 1998.

[6] Saverio Mascolo and Mario Gerla. TCP congestion avoidance using explicit buffer notification. Technical report, University of California, Los Angeles, California, February 1998.

[7] Rajesh Krishnan. TCP over load-reactive links. Technical Memorandum 1281, BBN, February 2001.

[8] O. Bonaventure. *Integration of ATM under TCP/IP to provide services with minimum guaranteed bandwidth*. PhD thesis, University of Liege, 1998.

[9] Kjersti Moldeklev and Per Gunningberg. How a large ATM MTU causes deadlocks in TCP data transfers. *IEEE/ACM Transactions on Networking*, 3(4):409–422, 1995.

[10] E. Chan, V. Lee, and J. Ng. On the performance of bandwidth allocation strategies for interconnecting ATM and connectionless networks. *ACM Computer Communications Review*, 26(1):29–38, January 1996.

[11] D. Clark and W. Fang. Explicit allocation of best-effort packet delivery service. *IEEE/ACM Transactions on Networking*, 6(4), August 1998.

[12] Rajesh Krishnan and James Sterbenz. TCP over load-reactive links. In *International Conference on Network Protocols (ICNP)*, 2001.

[13] Henrik Lundqvist. Performance evaluation for IP over DTM. Master's thesis, Linköping University, Linköping, 1998.

Chapter 8

Paper C: A Multi Path Routing Algorithm for IP Networks Based on Flow Optimisation

Henrik Abrahamsson, Bengt Ahlgren, Juan Alonso, Anders Andersson, and Per Kreuger. In *Proceedings of the Third International Workshop on Quality of Future Internet Services (QoFIS)*, Zürich, Switzerland, October 2002.

Abstract

Intra-domain routing in the Internet normally uses a single shortest path to forward packets towards a specific destination with no knowledge of traffic demand. We present an intra-domain routing algorithm based on multi-commodity flow optimisation which enable load sensitive forwarding over multiple paths. It is neither constrained by weight-tuning of legacy routing protocols, such as OSPF, nor requires a totally new forwarding mechanism, such as MPLS. These characteristics are accomplished by aggregating the traffic flows destined for the same egress into one commodity in the optimisation and using a hash based forwarding mechanism. The aggregation also results in a reduction of computational complexity which makes the algorithm feasible for on-line load balancing. Another contribution is the optimisation objective function which allows precise tuning of the tradeoff between load balancing and total network efficiency.

8.1 Introduction

As IP networks are becoming larger and more complex, the operators of these networks gain more and more interest in *traffic engineering* [1]. Traffic engineering encompasses performance evaluation and performance optimisation of operational IP networks. An important goal with traffic engineering is to use the available network resources more efficiently for different types of load patterns in order to provide a better and more reliable service to customers.

Current routing protocols in the Internet calculate the shortest path to a destination in some metric without knowing anything about the traffic demand or link load. Manual configuration by the network operator is therefore necessary to balance load between available alternate paths to avoid congestion. One way of simplifying the task of the operator and improve use of the available network resources is to make the routing protocol sensitive to traffic demand. Routing then becomes a flow optimisation problem.

One approach taken by others [2, 3, 4] is to let the flow optimisation result in a set of link weights that can be used by legacy routing protocols, e.g., open shortest path first (OSPF), possibly with equal cost multi-path (ECMP) forwarding. The advantage is that no changes are needed in the basic routing protocol or the forwarding mechanism. The disadvantage is that the optimisation is constrained by what can be achieved with tuning the weights. Another approach is to use MPLS [5], multi-protocol label switching, for forwarding traffic for large and long-lived flows. The advantage is that the optimisation is not constrained, but at the cost of more complexity in the routing and forwarding mechanisms.

Our goal is to design an optimising intra-domain routing protocol which is *not* constrained by weight-tuning, and which *can* be implemented with minor modifications of the legacy forwarding mechanism based on destination address prefix.

In this paper we present a routing algorithm for such a protocol based on multi-commodity flow optimisation which is both computationally tractable for on-line optimisation and also can be implemented with a near-legacy forwarding mechanism. The forwarding mechanism needs a modification similar to what is needed to handle the ECMP extension to OSPF.

The key to achieve this goal, and the main contribution of this paper, is in the modelling of the optimisation problem. We aggregate all traffic destined for a certain egress into one commodity in a multi-commodity flow optimisation. This reduces the number of commodities to at most N, the number of nodes, instead of being N^2 when the problem is modelled with one commodity for

each pair of ingress and egress nodes. As an example, the computation time for a 200 node network was in one experiment 35 seconds. It is this definition of a commodity that *both* makes the computation tractable, *and* the forwarding simple.

Another important contribution is the definition of an optimisation objective function which allows the network operator to choose a maximum desired link utilisation level. The optimisation will then find the most efficient solution, if it exists, satisfying the link level constraint. Our objective function thus enables the operator to control the trade-off between minimising the network utilisation and balancing load over multiple paths.

The rest of the paper is organised as follows. In the next section we describe the overall architecture where our optimising routing algorithm fits in. Section 8.3 presents the mathematical modelling of the optimisation problem. We continue with a short description of the forwarding mechanism in Sect. 8.4. After related work in Sect. 8.5 we conclude the paper.

8.2 Architecture

In this work we take the radical approach to completely replace the traditional intra-domain routing protocol with a protocol that is based on flow optimisation. This approach is perhaps not realistic when it comes to deployment in real networks in the near future, but it does have two advantages. First, it allows us to take full advantage of flow optimisation without being limited by current practise. Second, it results in a simpler overall solution compared to, e.g., the metric tuning approaches [2, 3, 4]. The purpose of taking this approach is to assess its feasibility and, hopefully, give an indication on how to balance flow optimisation functionality against compatibility with legacy routing protocols.

In this section we outline how the multi-commodity flow algorithm fits into a complete routing architecture. Figure 8.1 schematically illustrates its components. Flow measurements at all ingress nodes and the collection of the result are new components compared to legacy routing. The measurements continuously (at regular intervals) provide an estimate of the current demand matrix to the centralised flow optimisation. The demand matrix is aggregated at the level of all traffic from an ingress node destined for a certain egress node.

If a more fine-grained control over the traffic flows are desired, for instance to provide differentiated quality of service, a more fine-grained aggregation level can be chosen. This results in more commodities in the optimisation, which can be potential performance problem. One approach is to introduce two

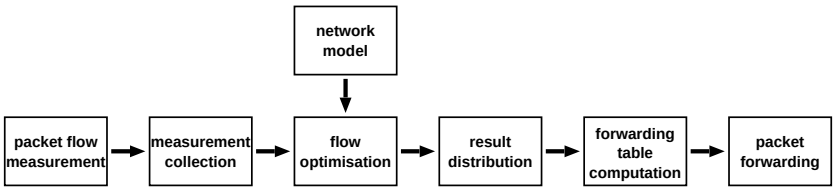


Figure 8.1: Routing architecture with flow optimisation.

levels in the optimisation, one with a longer time-scale for quality of service flows.

The demand matrix is input to the flow optimiser together with a model of the network. The result of the optimisation is a set of values y_{ij}^t, which encode how traffic arriving at a certain node (i), destined for a certain egress node (t) should be divided between the set of next hops (j). These values are used at each node together with a mapping between destination addresses and egress nodes to construct forwarding tables. Finally, the packet forwarding mechanism is modified to be able to distinguish packets destined for a certain egress node, and to forward along multiple paths toward those egresses.

The computation of the multi-commodity flow optimisation algorithm is inherently centralised. In this paper we also think of the computation as implemented in a central server. If a so-called bandwidth broker is needed or desired for providing a guaranteed quality of service, it is natural to co-locate it with optimisation. We however see the design of a distributed mechanism implementing flow optimisation as an important future work item.

The timescale of operation is important in an optimising routing architecture. There are several performance issues that put lower bounds on the cycle flow measurement–optimisation–new forwarding tables. The flow measurement need to be averaged over a long enough time to get sufficiently stable values. Our current research as well as others [6] indicate that the needed stability exists in real networks at the timescale of a few, maybe five to ten, minutes. Other performance issues are the collection of the flow measurements, the computation of the optimisation algorithm, and the distribution of the optimisation result. Our initial experiments indicate that a new optimisation cycle can be started in approximately each five minutes for typical intra-domain sizes.

An issue that we have identified is how to handle multiple egresses for a destination injected into the domain by BGP, the border gateway protocol. A straightforward way to solve this is to introduce additional virtual nodes in

the network to represent a common destination behind both egresses. This approach may however introduce a large number of additional nodes. This will need to be more carefully considered in the future.

8.3 Optimisation

The routing problem in a network consists in finding a path or multiple paths that send traffic through the network without exceeding the capacity of the links. When using optimisation to find such (multiple) paths, it is natural to model the traffic problem as a (linear) multi-commodity network flow problem (see, e.g., Ahuja et al. [7]), as many authors have done.

First, the network is modelled as a directed graph (this gives the topology, i.e., the static information of the traffic problem), and then the actual traffic situation (i.e., the dynamic part of the problem, consisting of the current traffic demand and link capacity) as a linear program. In modelling the network as a graph, a node is associated to each router and a directed edge to each directional link physically connecting the routers. Thus, we assume a given graph $G = (N, E)$, where N is a set of nodes and E is the set of (directed) edges. We will abuse language and make no distinction between graph and network, node and router, or edge and link.

Every edge $(i, j) \in E$ has an associated capacity k_{ij} reflecting the bandwidth available to the corresponding link. In addition, we assume a given *demand matrix* $D = D(s, t)$ expressing the traffic demand from node s to node t in the network. This information defines the routing problem. In order to formulate it as a multi-commodity flow (MCF) problem we must decide how to model commodities. In the usual approach [7, 2, 8] commodities are modelled as source-destination pairs that are interpreted as "all traffic from source to destination". Thus, the set of commodities is a subset of the Cartesian product $N \times N$; consequently, the number of commodities is bounded by the square of the number of nodes. To reduce the size of the problem and speed-up computations, we model instead commodities as (only destination) nodes, i.e., a commodity t is to be interpreted as "all traffic to t". Thus, our set of commodities is a subset of N and, hence, there are at most as many commodities as nodes. The corresponding MCF problem can be formulated as follows:

$$\min \{f(y) \mid y \in P_{12}\} \qquad (MCF_{12})$$

where $y = (y_{ij}^t),$ for $t \in N, (i, j) \in E$, and P_{12} is the polyhedron defined by

the equations:

$$\sum_{\{j|(i,j)\in E\}} y_{ij}^t - \sum_{\{j|(j,i)\in E\}} y_{ji}^t = d(i,t) \quad \forall i,t \in N \tag{8.1}$$

$$\sum_{t\in N} y_{ij}^t \leq k_{ij} \quad \forall (i,j) \in E \tag{8.2}$$

where

$$d(i,t) = \begin{cases} -\sum_{s\in N} D(s,t) & \text{if } i = t \\ D(i,t) & \text{if } i \neq t \end{cases} .$$

The variables y_{ij}^t denote the amount of traffic to t routed through the link (i,j). The equation set (1) state the condition that, at intermediate nodes i (i.e., at nodes different from t), the outgoing traffic equals the incoming traffic plus traffic created at i and destined to t, while at t the incoming traffic equals all traffic destined to t. The equation set (2) state the condition that the total traffic routed over a link cannot exceed the link's capacity.

It will also be of interest to consider the corresponding problem *without* requiring the presence of the equation set (2). We denote this problem (MCF_1). Notice that every point $y = (y_{ij}^t)$ in P_{12} or P_1 represents a possible solution to the routing problem: it gives a way to route traffic over the network so that the demand is met and capacity limits are respected (when it belongs to P_{12}), or the demand is met but capacity limits are not necessarily respected (when it belongs to P_1). Observe that $y = (0)$ is in P_{12} or in P_1 only in the trivial case when the demand matrix is zero.

A general linear objective function for either problem has the form $f(y) = \sum_{t,(i,j)} b_{ij}^t \; y_{ij}^t$. We will, however, consider only the case when all $b_{ij}^t = 1$ which corresponds to the case where all commodities have the same cost on all links. We will later use different objective functions (including non-linear ones) in order to find solutions with desired properties.

8.3.1 Desirable Solutions

In short, the solutions we consider to be desirable are those which are *efficient* and *balanced*. We make these notions precise as follows.

We use the objective function considered above, $f(y) = \sum_{t,(i,j)} y_{ij}^t$, as a measure of efficiency. Thus, given y_1, y_2 in P_{12} or P_1, we say that y_1 is *more efficient* than y_2 if $f(y_1) \leq f(y_2)$. To motivate this definition, note that

whenever traffic between two nodes can be routed over two different paths of unequal length, f will choose the shortest one. In case the capacity of the shortest path is not sufficient to send the requested traffic, f will utilise the shortest path to 100% of its capacity and send the remaining traffic over the longer path.

Given a point $y = (y_{ij}^t)$ as above, we let $Y_{i,j} = \sum_{t \in N} y_{ij}^t$ denote the total traffic sent through (i, j) by y. Every such y defines a *utilisation* of edges by the formula $u(y, i, j) = Y_{ij}/k_{ij}$, and $u(y, i, j) = 0$ when $k_{ij} = 0$. Let $u(y)$ denote the maximum value of $u(y, i, j)$ where (i, j) runs over all edges. Given an $\ell > 0$, we say that $y \in P_{12}$ (or $y \in P_1$) is *ℓ-balanced* if $u(y) \leq \ell$. For instance, a solution is (0.7)-balanced if it never uses any link to more than 70 % of its capacity.

8.3.2 How to Obtain Desirable Solutions

Poppe et al. [8] have proposed using different linear objective functions in order to obtain traffic solutions that are desirable with respect to several criteria (including balance, in the form of minimising the maximum utilisation of edges). Fortz and Thorup [2, 3], on the other hand, considers a fixed piece-wise linear objective function (consisting of six linear portions for each edge) which makes the cost of sending traffic along an edge depend on the utilisation of the edge. By making the cost increase drastically as the utilisation approaches 100 %, the function favours balanced solutions over congested ones. As the authors express it, their objective function "provides a general best effort measure".

Our contribution is related to the above mentioned work in that we use different objective functions to obtain desirable solutions, and the functions are piece-wise linear and depend on the utilisation. In contrast, our work defines different levels of balance (namely, ℓ-balance). For each such level, a simple piece-wise linear objective function consisting of two linear portions for each edge is *guaranteed* to find ℓ-balanced solutions provided, of course, that such solutions exist. Moreover, the solution found is guaranteed to be more efficient than any other ℓ-balanced solution.

Another distinctive feature of our functions is that they are defined through a uniform, theoretical "recipe" which is valid for every network. We thus eliminate the need to use experiments to adapt our definitions and results to each particular network. Finally, the fact that our functions consist of only two linear portions, shorten the execution time of the optimisation.

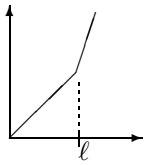


Figure 8.2: The link cost function $C^{\ell,\lambda}$.

8.3.3 The Result

To formulate our result we need to introduce some notation. Let $y = (y_{ij}^t)$ be a point of P_{12} or P_1, and suppose given real numbers $\lambda > 1$ and $\ell > 0$. We define the link cost function (illustrated in Fig. 8.2)

$$C^{\ell,\lambda}(U) = \begin{cases} U & \text{if } U \leq \ell \\ \lambda\ U + (1-\lambda)\ \ell & \text{if } U \geq \ell \end{cases} .$$

We use this function in the definition of the following objective function:

$$f^{\ell,\lambda}(y) = \sum_{(i,j)\in E} k_{ij}\ C^{\ell,\lambda}(u(y,i,j))$$

We also need to define the following constants:

$$v = \min\{f(y) \,|\, y \in P_{12}\} \qquad \text{and} \qquad V = \max\{f(y) \,|\, y \in P_{12}\}$$

Notice that $v > 0$ since $D(s,t) > 0$, and $V < \infty$ since the network is finite and we are enforcing the (finite) capacity conditions. At a more practical level, v can be computed by simply feeding the linear problem $\min\{f(y) \,|\, y \in P_{12}\}$ into CPLEX and solving it. Then, to compute V, one changes the same linear problem to a max problem (by replacing "min" by "max") and solves it.

Finally, let $\delta > 0$ denote the minimum capacity of the edges of positive capacity. We can now state the following theorem whose proof is given in a technical report [9]:

Theorem 1. *Let ℓ, ϵ be real numbers satisfying $0 < \ell < 1$ and $0 < \epsilon < 1 - \ell$. Suppose that $y \in P_1$ is ℓ-balanced, and let $\lambda > 1 + \frac{V^2}{v\delta\epsilon}$. Then any solution x of MCF_1 with objective function $f^{\ell,\lambda}$ is $(\ell+\epsilon)$-balanced. Moreover, x is more efficient than any other $(\ell+\epsilon)$-balanced point of P_1.*

Observe that, since $\ell < 1$ and $y \in P_1$ is ℓ-balanced, we can use MCF_1 instead of MCF_{12}. Informally, the theorem says that if there are ℓ-balanced

solutions, then $f^{\ell,\lambda}$ will find one. The number $\epsilon > 0$ is a technicality needed in the proof. Notice that it can be chosen arbitrarily small.

Theorem 1 can be used as follows. Given a target utilisation ℓ, say $\ell = 0.7$, compute $\frac{V^2}{v\delta\epsilon}$, choose a λ as in Theorem 1, and choose $\epsilon > 0$, say $\epsilon = 0.01$. Finally, compute a solution, say x, of MCF_1 with objective function $f^{\ell,\lambda}$. Then there are two exclusive possibilities: either x is 0.71-balanced or there is no such solution. In the last case, x can be thought of as a "best effort" solution since we have penalised all utilisation above 0.7 (which forces traffic using edges to more than 70 % of capacity to try to balance) but no 0.71-balanced solution exists. At this point we can either accept this best effort solution or iterate, this time setting the balance target to, say, 0.85, etc. After a few iterations we arrive at a solution which is "sufficiently" balanced or we know that there is no solution that is ℓ-balanced for the current value of ℓ which, we may decide, is so close to 1 that it is not worthwhile to continue iterating.

8.3.4 A Generalisation

Theorem 1 has a useful generalisation that can be described as follows. Partition the set of edges E into a family (E_i) of subsets, and choose a target utilisation ℓ_i for each E_i. The generalised theorem says that for small $\epsilon > 0$ we can define a function corresponding to $f^{\ell,\lambda}$ in Theorem 1, such that solving MCF_1 with this objective function will result in efficient solutions that are $(\ell_i + \epsilon)$-balanced on E_i provided, of course, that such solutions exist. The generalised theorem is more flexible in that it allows us to seek solutions with different utilisation in different parts of the network.

8.3.5 Quantitative Results

We have used CPLEX 7.1[1] on a Pentium laptop to conduct numerical experiments with a graph representing a simplified version of a real projected network. The graph has approximately 200 nodes and 720 directed edges. If we had modelled MCF with source-destination pairs as commodities, the linear problem corresponding to MCF_{12} would consist of some 8 million equations and 30 million variables. Modelling commodities as traffic to a node, MCF_{12} contains, in contrast, "only" about 40 000 constraints and 140 000 variables. Solving MCF_1 with objective function $f^{\ell,\lambda}$ takes approximately 35 seconds.

Solving the same problem with the objective function considered by Fortz and Thorup [2, 3] takes approximately 65 seconds. Our experiments suggest

[1] ILOG CPLEX 7.1 http://www.ilog.com

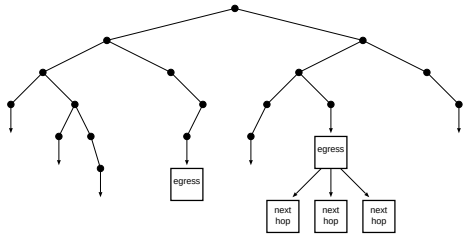


Figure 8.3: Address lookup data structure for multiple path forwarding.

that this function picks solutions that minimise balance. In contrast, with $f^{\ell,\lambda}$ we can choose any desired level of balance (above the minimum, of course).

8.4 Multi-Path Forwarding

By modelling the routing problem as "all traffic to t", as described in the previous section, we get an output from the optimisation that is well suited for packet forwarding in the routers. The result from the optimisation, the y_{ij}^t values, tells how packets at a certain node (i) to a certain egress node (t) in the network should be divided between the set of next hops (j). We thus need a forwarding mechanism that can distinguish packets destined for a certain egress, and that can forward along multiple paths.

To enable forwarding along multiple paths, we introduce one more step in the usual forwarding process. An egress data structure is inserted in the address lookup tree just above the next hop data structure as illustrated in Fig. 8.3. A longest prefix match is done in the same manner as in a standard forwarding table, except that it results in the destination egress node. The egress data structure stores references to the set of next hops to which traffic for that egress should be forwarded, as well as the desired ratios (the y_{ij}^t for all js) between the next hops.

In order to populate the forwarding tables a mapping has to be created between destination addresses and egress nodes. The needed information is the same as a regular intra-domain routing protocol needs, and is obtained in much the same way. For destinations in networks run by other operators (i.e., in other routing domains), the mapping is obtained from the BGP routing protocol. For intra-domain destinations, the destination prefix is directly connected to the egress node.

Mechanisms for distributing traffic between multiple links have been thoroughly evaluated by Cao et al. [10]. We propose to use a table based hashing mechanism with adaptation, because it can distribute the load according to unequal ratios, is simple to compute, and adapts to the properties of the actual traffic.

Similar mechanisms already exist in commercial routers in order to handle the equal cost multi-path extension to OSPF and similar protocols.

8.5 Related Work

With the prospect of better utilising available network resources and optimising traffic performance, a lot of research activity is currently going on in the area of traffic engineering. The general principles and requirements for traffic engineering are described in the RFC 3272 [1] produced by the IETF Internet Traffic Engineering working group. The requirements for traffic engineering over MPLS are described in RFC 2702 [5].

Several researchers use multi-commodity flow models in the context of traffic engineering. Fortz and Thorup [2, 3] use a local search heuristics for optimising the weight setting in OSPF. They use the result of multi-commodity flow optimisation as a benchmark to see how close to optimal the OSPF routing can get using different sets of weights. Mitra and Ramakrishnan [11] describes techniques for optimisation subject to QoS constraints in MPLS-supported IP networks. Poppe et al. [8] investigate models with different objectives for calculating explicit routes for MPLS traffic trunks. Multi-commodity flow and network flow models in general have numerous application areas. A comprehensive introduction to network flows can be found in Ahuja et al. [7].

A somewhat controversial assumption when using multi-commodity flow optimisation is that an estimate of the demand matrix is available. The problem of deriving the demand matrix for operational IP networks is considered by Feldmann et al. [12]. The demand matrix only describes the current traffic situation but, for an optimisation to work well, it must also be a good prediction of the near future. Current research in traffic analysis by Bhattacharyya et al. [6] and Feldmann et al. [12] indicate that sufficient long term flow stability exists on backbone links in timescales of minutes and hours and in manageable aggregation levels to make optimisation feasible.

8.6 Conclusions

We have taken the first steps to introduce flow optimisation as a routing mechanism for an intra-domain routing protocol. We have presented a routing algorithm based on multi-commodity flow optimisation which we claim is computationally tractable for on-line routing decisions and also only require a small modification to the legacy packet forwarding mechanism. More work is however needed on other components in order to design and implement a complete routing protocol using our algorithm.

The key issue, and our main contribution, is the mathematical modelling of commodities. Traffic destined for a certain egress node is aggregated into a single commodity. This results in computational requirements an order of magnitude smaller than in the traditional models where the problem is modelled with one commodity for each flow from one ingress to one egress node.

Multi-path forwarding of the aggregates produced by the optimiser is then handled by a hash based forwarding mechanism very similar to what is needed for OSPF with ECMP.

Another contribution is the design of a generic objective function for the optimisation which allows the network operator to choose a desired limit on link utilisation. The optimisation mechanism then computes a most efficient solution given this requirement, when possible, and produces a best effort solution in other cases. The process can be iterated with, e.g., binary search to find a feasible level of load balance for a given network load.

Bibliography

[1] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. Overview and principles of Internet traffic engineering. Internet RFC 3272, May 2002.

[2] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *Proceedings IEEE INFOCOM 2000*, pages 519–528, Israel, March 2000.

[3] B. Fortz and M. Thorup. Optimizing OSPF/IS-IS weights in a changing world. *IEEE Journal on Selected Areas in Communications*, 20(4):756–767, May 2002.

[4] D. Yuan. *Optimization Models and Methods for Communication Network Design and Routing*. PhD thesis, Linkpings Universitet, 2001.

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

[6] S. Bhattacharyya, C. Diot, J. Jetcheva, and N. Taft. Pop-level and access-link-level traffic dynamics in a tier-1 PoP. In *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, San Francisco, USA, November 2001.

[7] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows*. Prentice-Hall, 1993.

[8] F. Poppe, S. van den Bosch, P. de la Vallée-Poussin, H. van Hove, H. de Neve, and G. Petit. Choosing the objectives for traffic engineering in IP backbone networks based on Quality-of-Service requirements. In *Proceedings of First COST 263 International Workshop, QofIS*, pages 129–140, Berlin, Germany, September 2000.

[9] J. Alonso, H. Abrahamsson, B. Ahlgren, A. Andersson, and P. Kreuger. Objective functions for balance in traffic engineering. Technical Report T2002:05, SICS – Swedish Institute of Computer Science, May 2002.

[10] Z. Cao, Z. Wang, and E. Zegura. Performance of hashing-based schemes for internet load balancing. In *Proceedings of IEEE INFOCOM 2000*, Israel, March 2000.

[11] D. Mitra and K. G. Ramakrishnan. A Case Study of Multiservice, Multipriority Traffic Engineering Design for Data Networks. In *Proceedings of Globecom'99*, Brazil, 1999.

[12] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True. Deriving traffic demands for operational IP networks: Methodology and experience. In *Proceedings of ACM SIGCOMM'00*, Stockholm, Sweden, August 2000.

Chapter 9

Paper D:
Traffic Engineering in Ambient Networks: Challenges and Approaches

Henrik Abrahamsson and Anders Gunnar.
In *Proceedings of Second Swedish National Computer Networking Workshop (SNCNW)*, 2004, Karlstad, Sweden.

Abstract

The focus of this paper is on traffic engineering in ambient networks. We describe and categorize different alternatives for making the routing more adaptive to the current traffic situation and discuss the challenges that ambient networks pose on traffic engineering methods. One of the main objectives of traffic engineering is to avoid congestion by controlling and optimising the routing function, or in short, to put the traffic where the capacity is. The main challenge for traffic engineering in ambient networks is to cope with the dynamics of both topology and traffic demands. Mechanisms are needed that can handle traffic load dynamics in scenarios with sudden changes in traffic demand and dynamically distribute traffic to benefit from available resources. Tradeoffs between optimality, stability and signaling overhead that are important for traffic engineering methods in the fixed Internet becomes even more critical in a dynamic ambient environment.

9.1 Introduction

The existing mobile and wireless link layer technologies like WLAN, GSM, 3G, etc, lack a common control plane in order to enable end-users to benefit fully from the offered access connectivity. For instance, operators only grant access to users with whom they have previously signed an agreement. Similarly, there is no technology to automatically and transparently select the best and most cost effective link technology for the end-user. The Ambient Networks project [1] aims to address these issues and to provide an affordable, robust and technology independent communication platform beyond 3G. Ambient networks also support cooperation between operators to handle control functions such as managing mobility, security, and quality of service. The key concept of ambient networks is network composition. Networks establish inter-network agreements on-demand without human interaction. Network composition will provide access to any network instantly anywhere at any time.

Instant network composition brings new challenges to traffic engineering and monitoring of the network. Traffic engineering encompasses performance evaluation and performance optimization of operational networks. An important goal is to avoid congestion in the network and to make better use of available network resources by adapting the routing to the current traffic situation. More efficient operation of a network means more traffic can be handled with the same resources which enables a more affordable service. As ambient networks compose and decompose the topology and traffic patterns can change rapidly. This means that one can not rely only on long-term network planning and dimensioning that are done when the network is first built. Traffic engineering mechanisms are needed to adapt to changes in topology and traffic demand and dynamically distribute traffic to benefit from available resources.

In this paper we identify and analyse the challenges ambient networks pose to traffic engineering. At this stage, we intend to identify research issues and discuss how we intend to address them. Consequently, we do not aim to provide integrated solutions to the problems identified.

The rest of the paper is organized as follows. In the next section we introduce Ambient Networks. In the following section we give a short introduction to traffic engineering. Section 4 discuss the challenges and research issues for traffic engineering in Ambient Networks. Finally, in the last section we give a short summary and discussion.

9.2 Ambient Networks

The Ambient Networks project [1], started in 2004, is an integrated project within the EU's 6th Framework Programme. The overall purpose of the project is to build an architecture for mobile communication networks beyond 3G [2]. Ambient networks represents a new networking concept which aims to enable the cooperation of heterogeneous networks belonging to different operators or technology domains.

The basis for communication in Ambient Networks is IP. However, the architecture should overcome the diversity in access network technologies. To be specific, Ambient Networks should support present access technologies as well as enable incremental introduction of new access technologies and services to the communication architecture. Further, the project aims to enable cooperation between operators to handle control functions such as managing mobility, security, and quality of service.

A key concept in ambient networks is network composition. The vision is to allow agreement for cooperation between networks on demand and without the need of preconfiguration or offline negotiation between network operators. The composition should also be rapid enough to handle adaptation to moving networks such as a train with an internal access network passing through an operators network. This instant network composition brings new challenges to network management and traffic engineering in ambient networks [3].

In conventional IP backbone networks the variability both in traffic patterns as well as in topology is small. The network topology only changes if routers or links go up/down or when new links are added to the network. Internet traffic has been shown to have very bursty and self-similar behaviour on short time-scales but if we consider timescales of tens of minutes the variability in traffic basically follow diurnal patterns in a highly predictable manner. In Ambient Networks on the other hand, network topology and traffic patterns is expected to be under constant change as networks compose and de-compose. This is further illustrated in Figure 9.1. The figure shows variability in traffic patterns along the x-axis and variability in topology along the y-axis. To some extent the characteristics of Ambient Networks overlap the characteristics of conventional IP networks. However, Ambient Networks cover a much broader spectrum of variability in both topology and traffic patterns.

In ambient networks we can expect both conditions similar to current IP backbone networking as well as conditions where the topology changes are similar to ad-hoc networks and traffic demands shift due to mobility of networks and network composition. However, this paper is focused on traffic

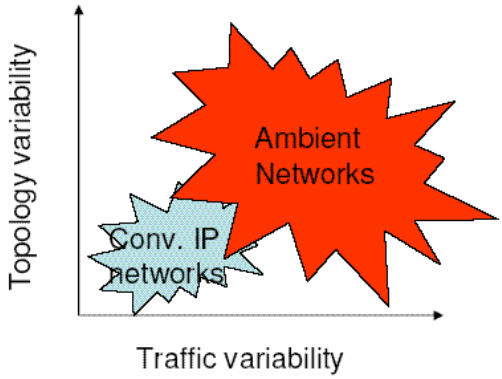


Figure 9.1: Characteristics of Ambient Networks compared to conventional IP networks.

engineering under varying traffic patterns. The behaviour of network topology is considered to be similar to the conditions in conventional IP networks.

9.3 Traffic Engineering

For a network operator it is important to analyse and tune the performance of the network in order to make the best use of it. The process of performance evaluation and optimization of operational IP-networks is often referred to as traffic engineering. One of the major objectives is to avoid congestion by controlling and optimizing the routing function. The traffic engineering process can be divided in three parts as illustrated in Figure 9.2. The first step is the collection of necessary information about network state. To be specific, the current traffic situation and network topology. The second step is the optimisation calculations. And finally, the third step is the mapping from optimization to routing parameters. Current routing protocols are designed to be simple and robust rather than to optimize the resource usage. The two most common intra-domain routing protocols today are OSPF (Open Shortest Path First) and IS-IS (Intermediate System to Intermediate System). They are both link-state protocols and the routing decisions are typically based on link costs and a shortest (least-cost) path calculation. While this approach is simple, highly distributed

and scalable these protocols do not consider network utilization and do not always make good use of network resources. The traffic is routed on the shortest path through the network even if the shortest path is overloaded and there exist alternative paths. With an extension to the routing protocols like equal-cost multi-path (ECMP) the traffic can be distributed over several paths but the basic problems remain. An underutilized longer path cannot be used and every equal cost path will have an equal share of load.

Figure 9.2: The traffic engineering process.

This section introduces and analyses different approaches to traffic engineering in IP networks. In the next subsection we present a framework to categorize different methods of traffic engineering. This framework is used in the following section to analyse a selection of suggested methods for traffic engineering.

9.3.1 Classification of Traffic Engineering Methods

A classification of traffic engineering schemes is possible along numerous axis. Our framework is intended to facilitate the analysis and help us identify the requirements for traffic engineering in Ambient Networks.

- **Optimize legacy routing vs novel routing mechanisms**. One approach is to optimize legacy routing protocols. The advantage is easy deployment of the traffic engineering mechanism. However, the disadvantage is the constraints imposed by legacy routing.
- **Centralized vs distributed solutions**. A centralized solution is often simpler and less complex than a distributed, but is more vulnerable than a distributed solution.
- **Local vs global information**. Global information of the current traffic situation enables the traffic engineering mechanism to find a global optimum for the load balancing. The downside is the signaling required

to collect the information. In addition, in a dynamic environment, the information quickly becomes obsolete.

- **Off-line vs on-line traffic engineering**. Off-line traffic engineering is intended to support the operator in the management and planning of the network. On-line traffic engineering on the other hand, reacts to a signal from the network and perform some action to remedy the problem.

The taxonomy above is intended to assist us in the analysis of traffic engineering methods in Ambient Networks and should not be regarded as complete. A detailed taxonomy of traffic engineering methods can be found in RFC 3272 [4].

9.3.2 Previous Work

The general problem of finding the best way to route traffic through a network can be mathematically formulated as a multi-commodity flow (MCF) optimization problem. This has recently been used by several research groups to address traffic engineering problems [5, 6, 7, 8, 9]. In the simplest case the optimization result can be used as just a benchmark when evaluating the performance of the network to see how far from optimal the current routing is. A number of attempts has been made to optimize legacy routing protocols [6, 8, 9]. Fortz *et.al* [6] uses a search heuristic to optimize the OSPF link weights to balance load in a network and the MCF optimization serves as a benchmark for the search heuristic. Similarly, Wang *et.al* attempts to find the optimal link weights for OSPF routing. However, they formulate the problem as a linear program and find the link weights by solving the dual problem. The optimization can also be used as a basis for allocating Label Switch Paths (LSP) in MPLS [7, 10]. A more long-term research goal would be to construct a new multi-path routing protocol based on flow optimization [5]. A somewhat different approach is taken by Sridharan *et.al* [8]. Instead of calculating the link weights the authors use a heuristic to allocate routing prefixes to equal-cost multi-paths. Again the MCF optimization serves as a benchmark for the heuristic.

All global optimization methods require an estimate of the current traffic situation as input to the estimation. The current traffic situation can be succinctly captured in a traffic matrix that has one entry for each origin-to-destination traffic demand. However, the support in routers to measure the traffic matrix is only rudimentary. Instead operators are forced to estimate the traffic matrix from incomplete data. This estimation problem has recently been

addressed by many researcher. An evaluation of a wide selection of estimation methods and further references can be found in Gunnar *et.al* [11].

An attempt to localize and distribute the routing decisions is Adaptive Multi-path routing (AMP) [12]. In AMP information on the traffic situation on links is only distributed to the immediate neighbors of each router. Hence, AMP relies on local information in neighboring routers to calculate next hop towards the destination. Andres-Colas *et.al* [13] introduces Multi-Path Routing with Dynamic Variance (MRDV), where load on the next hop towards the destination is included in the selection of next hop towards the destination. In this approach no load information is exchanged between routers. Instead the cost of each path towards the destination is weighted by a variance factor which reflect load on the next hop. Hence, traffic is shifted from heavily loaded links to links with less load. A related approach is introduced by Vutukury *et.al* [14]. Here the routing decision is divided into two steps. First, multiple loop-free paths are established using long term delay information. In the second step the routing parameters along the precomputed paths are adjusted using only local short-term delay information.

9.4 Challenges for Traffic Engineering in Ambient Networks

The main challenge for traffic engineering in Ambient Networks is to cope with the dynamics of both topology and traffic demands. Mechanisms are needed that can handle traffic load dynamics in scenarios with sudden changes in traffic demand and dynamically distribute traffic to benefit from available resources. As described in section 9.3.1 , different traffic engineering methods can be categorized by how much network state information they use. This ranges from methods that only use local state information to improve the load-balancing to optimization methods that need global state information in the form of link capacities and a traffic matrix as input. The trade-offs between optimality, stability and signaling overhead are crucial for traffic engineering methods in the fixed Internet and it is even more critical in a dynamic ambient environment.

The traffic engineering problem can best be modeled as a multi-commodity flow optimisation problem. This type of optimisation techniques take as input global information about the network state (i.e., traffic demands and link capacities) and can calculate the global optimal solution. In practice though, there might be several reasons why we need to deviate from the optimal use of the network. This could be because the calculations are too resource consuming

and take too long time. It could also be because the input needed is hard to measure and collect and that it varies too much over time so it would create too much signaling overhead or create instabilities.

MCF optimisation problems easily becomes large with tens of thousands of variables and constraints. But it is possible to calculate the global optimal solution in tens of seconds even for large networks [5] if no constraints are given on the number of paths that can be used. Finding the optimal set of weights in OSPF though usually has to rely on heuristic methods.

One can argue that, if it is important to make the best possible use of network resources then the routing should not be restricted to what can be achieved by tuning the weights in the legacy routing protocols. Instead, the optimisation should come first and the result should be implemented using new routing mechanisms if needed. On the other hand, the study by Fortz *et.al* [6] shows that in practice the solutions that can be achieved by proper weight settings in OSPF are close to the optimal at least for the networks they investigated.

Multi-commodity flow optimization as well as heuristic methods for setting optimal weights in OSPF are both typical examples of centralised schemes that use global information in the form of topology and traffic matrix and produce global optimum routing or at least results that are good for the network as a whole. The problems with this type of solution is measuring the traffic demands that are needed as input and the signaling overhead created when collecting this data. A centralised solution also creates a possible bottleneck and a single point of failure. Further, in a dynamic environment the traffic data quickly becomes obsolete. If the routing decisions are based on the wrong input we may create congestion that would not be there if just shortest-path routing had been used. This sensitivity to the traffic dynamics of course holds for all types of load-sensitive routing.

Examples of other schemes that uses global information about both the topology and the traffic situation but takes local decisions (and so avoids some of the problems with a centralised solution) is different kinds of QoS-routing schemes. Here information about for instance delay or load on each link in the network is flooded to all nodes. Each node then makes shortest-path (or least-cost) calculations in this metric. Each node chooses the best paths through the network from its own perspective but the decisions are all local decisions without consideration of the network as a whole. So care must be taken with this type of mechanism to avoid hot-spots where everybody moves traffic to underutilised links and route flapping were nodes constantly shift load back and forth.

Another possibility would be to only use local information when taking local decisions and so avoid all the signaling overhead [13]. If we can assume that the topology is much more constant than the traffic load then we can use global information about the topology i.e using legacy protocols like OSPF to calculate the connectivity (shortest paths) and use only local information about the traffic situation to balance the load in the network. This is an interesting approach in a dynamic environment such as ambient networks, with sudden changes in traffic demand. For instance in a scenario with a moving network such as a train with an internal access network passing through an operators network. Instead of flooding the network with load information and wait for a new routing to be calculated a node can make local decisions and adapt to the situation. A node that experiences a sudden increase in traffic demand can directly shift load from heavily loaded links to underutilised paths. The drawback of this is of course that the consequences of the local decisions for the network as a whole are difficult to grasp. Care must be taken so that local improvements don't create overload somewhere else in the network. So, a careful evaluation of this type of mechanism is needed.

There are different timescales for traffic engineering. An interesting approach would be if global information reflecting the traffic situation in a coarser and longer time perspective could be used to make a tentative routing calculation for the whole network. And let the nodes fine-tune the routing parameters with respect to local information in the nodes or information gained from the immediate vicinity of respective node. But this is a topic for further study.

9.5 Summary

This paper identifies the requirements and challenges for traffic engineering in a dynamic environment. We give a short introduction to the Ambient Networks project which aims to provide a novel mobile communication platform beyond 3G. Further, a framework for classification of traffic engineering methods is introduced to facilitate the analysis and identification of challenges for traffic engineering in Ambient Networks. This framework is used to discuss the properties a traffic engineering scheme must hold in order to meet the requirements of Ambient Networks.

9.6 Acknowledgments

This paper describes work undertaken in the context of the Ambient Networks - Information Society Technologies project, which is partially funded by the Commission of the European Union. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the Ambient Networks Project.

Bibliography

[1] WWI-AN Ambient Networks Project WWW Server. http://www.ambient-networks.org.

[2] N. Niebert, A. Schieder, H. Abramowicz, G. Malmgren, J. Sachs, U. Horn, C. Prehofer, and H. Karl. Ambient Networks: An Architecture for Communication Networks Beyond 3G. *IEEE Wireless Communications*, 11(2):14–22, April 2004.

[3] M. Brunner, A. Galis, L. Cheng, J. Andrs Cols, B. Ahlgren, A. Gunnar, H. Abrahamsson, R. Szabo, S. Csaba, J. Nielsen, A. Gonzalez Prieto, R. Stadler, and G. Molnar. Ambient Networks Management Challenges and Approaches. In *IEEE MATA 2004 1st International Workshop on Mobility Aware Technologies and Applications*, Florianopolis, Brazil, October 2004.

[4] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. Overview and principles of Internet traffic engineering. Internet RFC 3272, May 2002.

[5] H. Abrahamsson, J. Alonso, B. Ahlgren, A. Andersson, and P. Kreuger. A multi path routing algorithm for IP networks based on flow optimisation. In *Proceedings of the Third International Workshop on Quality of Future Internet Services (QoFIS)*, Zürich, Switzerland, October 2002.

[6] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *Proceedings IEEE INFOCOM 2000*, pages 519–528, Israel, March 2000.

[7] D. Mitra and K. G. Ramakrishnan. A Case Study of Multiservice, Multipriority Traffic Engineering Design for Data Networks. In *Proceedings of Globecom'99*, Brazil, 1999.

[8] A. Sridharan, R. Guérin, and C. Diot. Achieving Near-Optimal Traffic Engineering Solutions for Current OSPF/IS-IS Networks. In *IEEE Infocom*, San Francisco, March 2003.

[9] Y. Wang, Z. Wang, and L. Zhang. Internet Traffic Engineering without Full Mesh Overlaying. In *Proceedings of IEEE INFOCOM 2001*, Anchorage, Alaska, 2001.

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

[11] A. Gunnar, M. Johansson, and T. Telkamp. Traffic Matrix Estimation on a Large IP Backbone - a Comparison on Real Data. In *Proceedings of ACM Internet Measurement Conference*, Taormina, Sicily, Italy, October 2004.

[12] I. Gojmerac, T. Ziegler, and P. Reichel. Adaptive Multipath Routing Based on Local Distribution of Link Load Information. In *Proceedings of QofIS 2003*, pages 122–131, Stockholm, Sweden, Oct 2003.

[13] J. Andres-Colas, F. J. Ramon-Salguero, A. Molins-Jimenez, and J. Enriquez-Gabeiras. Multipath Routing with Dynamic Variance. COST 279 Technical Report TD02043, 2002.

[14] S. Vutukury and J.J. Garcia-Luna-Aceves. A Simple Approximation to Minimum Delay Routing. In *Proceedings of ACM SIGCOMM'99*, Cambridge, Massachusetts, September 1999.

Chapter 10

Paper E:
Robust Traffic Engineering using L-balanced Weight-Settings in OSPF/IS-IS

Henrik Abrahamsson and Mats Björkman.
Submitted for publication, October 2008.

Abstract

Internet traffic volumes continue to grow at a great rate, now pushed by video and TV distribution in the networks. This brings up the need for traffic engineering mechanisms to better control the traffic. The objective of traffic engineering is to avoid congestion in the network and make good use of available resources by controlling and optimising the routing function. The challenge for traffic engineering in IP networks is to cope with the dynamics of Internet traffic demands. Today, the main alternative for intra-domain traffic engineering in IP networks is to use different methods for setting the weights in the routing protocols OSPF and IS-IS. In this paper we revisit the weight setting approach to traffic engineering but with focus on robustness. We propose *l*-balanced weight settings that route the traffic on the shortest paths possible but make sure that no link is utilised to more than a given level *l*. This gives efficient routing of traffic and controlled spare capacity to handle unpredictable changes in traffic. We present a heuristic search method for finding *l*-balanced weight settings and show that it works well in real network scenarios.

10.1 Introduction

Internet traffic volumes continue to grow at a great rate, now pushed on by video and TV distribution in the networks. Increasing traffic volumes necessitate upgrades of network equipment and new investments for operators, and keep up-to-date the question of over-dimensioning network capacity versus using traffic engineering mechanisms for better handling the traffic. In addition, as new bandwidth demanding and also delay and loss sensitive services are introduced, it is even more important for the operator to manage the traffic situation in the network.

The main challenge for traffic engineering is to cope with the dynamics of traffic demands and topology. How to best model and describe aggregated Internet traffic is still an open area of research. On short timescales up to seconds the traffic is very bursty and on long timescales there are often predictable daily and weekly cycles. In between there can be unpredictable changes and shifts in traffic demand, for instance due to hotspots and flash crowds, or because a link goes down, there are changes in the inter-domain BGP routing, or because traffic in an overlay is re-directed. For future networks more variability in traffic demands is also expected due to mobility of nodes and networks and more dynamic on-demand service level agreements (SLA:s).

The traffic variability means that, even if we could measure the current traffic situation exactly, it would not always correctly predict the near future traffic situation and this needs to be taken into account when doing traffic engineering. Network operators often handle this by relying on simple well-tried techniques (like OSPF and IS-IS routing), over-dimensioning of network capacity, and simple rules of thumb (i.e upgrade the link capacity when mean utilisation reaches 70-80%) rather than introducing complex traffic engineering techniques.

In this paper we take this need for spare capacity and simple rules of thumb as our starting point. We revisit the approach of using weight settings in OSPF/IS-IS for traffic engineering but now with focus on robustness. We propose weight settings that we call *l-balanced* where the operator, by setting the parameter l (to say 80%), control the maximum utilisation level in the network and how much spare capacity is needed to handle unpredictable traffic changes. With an l-balanced routing the traffic takes the shortest paths possible but makes sure that no link is utilised to more than a given level l, if possible.

The main contributions in this paper are:

- We propose l-balanced weight settings in OSPF/IS-IS for robust traffic engineering.

- We present a heuristic search method for finding *l*-balanced weight settings and show that it works well in real network scenarios.
- We evaluate *l*-balanced routing and compare it with other proposed traffic engineering objectives for several real network topologies and traffic data sets.

If traffic levels continue to grow then of course network capacity needs to be added at some point. But traffic engineering with *l*-balanced routing can extend the upgrade cycle and postpone the investment, or be applied to better use the existing resources in the network until the highly utilised links have been upgraded.

The paper is organized as follows. Section 10.2 gives a short introduction to traffic engineering in IP networks and Section 10.3 discusses related work. We then present the *l*-balanced cost function in Section 10.4 and describe the search heuristic used for finding *l*-balanced weight settings. In Section 10.5 we evaluate the proposed methods. We show that the search heuristic works well for finding *l*-balanced weight settings in real traffic scenarios. Further, we compare the robustness of different weight-setting methods and investigate what happens to link utilisations in the network if a traffic demand suddenly increases. Finally, in Section 10.6 we make some concluding remarks about our findings.

10.2 Traffic Engineering in IP networks

The objective of traffic engineering is to avoid congestion in the network and to make better use of available network resources by adapting the routing to the current traffic situation. The traffic demands in a network changes over time and for network operators it is important to tune the network in order to accommodate more traffic and meet service level agreements (SLAs) made with their customers. This means that a network operator can not rely only on long-term network planning and dimensioning that are done when the network is first built. Robust traffic engineering mechanisms are needed that can adapt to changes in traffic demand and distribute traffic to benefit from available resources.

The first step in the traffic engineering process is to collect the necessary information about network topology and the current traffic situation. Most traffic engineering methods need as input a traffic matrix describing the demand between each pair of nodes in the network. The traffic matrix is then used as

input to the routing optimization step, and the optimized parameters are finally used to update the current routing.

Today, the main alternative for intra-domain traffic engineering in IP networks is to use different methods for setting the weights (and so decide upon the shortest paths) in the routing protocols OSPF (Open Shortest Path First) and IS-IS (Intermediate System to Intermediate System). These are both link-state protocols and the routing decisions are based on link costs and a shortest (least-cost) path calculation. With the equal-cost multi-path (ECMP) extension to the routing protocols the traffic can also be distributed over several paths that have the same cost. These routing protocols were designed to be simple and robust rather than to optimize the resource usage. They do not by themselves consider network utilisation and do not always make good use of network resources. The traffic is routed on the shortest path through the network even if the shortest path is overloaded and there exist alternative paths. It is up to the operator to find a set of link costs (weights) that is best suited for the current traffic situation and that avoids congestion in the network.

The general problem of finding the best way to route traffic through a network can be mathematically formulated as a multi-commodity flow (MCF) optimization problem (see, e.g., [1, 2, 3]). The network is then modeled as a graph. The problem consists of routing the traffic, given by a demand matrix, in the graph with given link capacities while minimizing a cost function. With no limitations on how the traffic flows can be divided over the network links the MCF optimal routing problem can be formulated and efficiently solved as a linear program. Introducing integer weights and ECMP shortest paths constraints, where the traffic no longer can be split arbitrarily, makes the problem computationally much harder. For reasonably sized networks one usually has to rely on search heuristics for determining the set of weights, rather than calculating the optimal weights.

10.3 Related work

Traffic engineering by finding a suitable set of weights in OSPF/IS-IS is a well studied area of research and it is described in recent textbooks in the area [3, 4]. When we now revisit the weight setting approach to traffic engineering we are most inspired by the pioneering works by Fortz and Thorup [2, 5] and Ramakrishnan and Rodrigues [6], in that we use a piece-wise linear cost function and search heuristics to find suitable weight settings.

Several studies [2, 7, 8, 9] have shown that even though we limit the rout-

ing of traffic to what can be achieved with weight-based ECMP shortest paths, and not necessarily the optimal weights but those found by search heuristics, it often comes close to the optimal routing for real network scenarios. How the traffic is distributed in the network very much depends on the objectives, usually expressed as a cost function, in the optimisation. An often proposed objective function is described by Fortz and Thorup [2] (and we will refer to it as the FT cost function further on). Here the sum of the cost over all links is considered and a piece-wise linear increasing cost function is applied to the flow on each link. The basic idea is that it should be cheap to use a link with small utilization while using a link that approaches 100% utilisation should be heavily penalized. The *l*-balanced cost function [1, 10] used in this paper is similar in that it uses a piecewise linear cost function to obtain desireable solutions. Additionaly, it gives the operator the opportunity to set the maximum wanted link utilisation. Cost functions for traffic engineering is further investigated by Balon *et.al* [11]

The main difference in this paper compared to previous work on weight settings is our focus on robustness and the objective of achieving a controlled spare capacity for handling unpredictable traffic shifts. For robust traffic engineering much of the focus is on handling multiple traffic matrices and traffic scenarios [5, 12, 13, 14, 15] and handling the trade-off between optimizing for the common case or for the worst case.

Xu *et.al* [16] describe a method to jointly solve the flow optimization and the link-weight approximation using a single formulation resulting in a more efficient computation. Their method can also direct traffic over non-shortest paths with arbitrary percentages. Their results should also be directly applicable to our problem of providing robustness to changes, by just substituting their piece-wise linear cost function with our cost function. In a continuation on this work Xu *et.al* [17] propose a new link-state routing protocol. The protocol splits traffic over multiple paths with an exponential penalty on longer paths and achieves optimal traffic engineering while retaining the simplicity of hop-by-hop forwarding.

10.4 L-balanced solutions

10.4.1 Optimal l-balanced routing

A routing is said to be *l-balanced* if the utilisation is less than or equal to l on every link in the network. For instance a solution is (0.7)-balanced if it never

uses any link to more than 70% of its capacity.

The *l*-balanced cost function, its theoretical foundation, and use in MCF optimisation is described in [1, 10]. The idea is to use a simple piece-wise linear cost function as shown in Figure 10.1 and apply it to the utilisation of each link in the network. The cost function consists of two linear portions where the slope of the second line segment should be large enough to penalise utilisation above *l* and balance traffic over longer paths.

The work in [1, 10] present a formula to calculate the cost function, for a given network topology and traffic situation, that guarantees to find a *l*-balanced optimal routing (provided, of course, that such solutions exist) that takes the shortest paths possible and makes sure that no link is utilised to more than *l*.

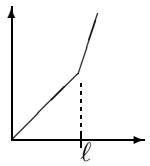


Figure 10.1: The link cost function.

10.4.2 Search for l-balanced weight settings

To apply the *l*-balanced routing in real OSPF/IS-IS networks we need to find *l*-balanced weight settings. For weight settings we dont have the guarantee to find an *l*-balanced routing in the same way as described for optimal routing above. But we want to use the *l*-balanced cost function to find weights settings that achieve the same effect of taking the shortest paths possible while routing the traffic so that no link is utilised to more than a given level *l*.

The optimal weights are often too computationally hard and time consuming to calculate for real networks and traffic scenarios. Instead we use a problem specific local search heuristic to determine the set of weights. An overview of local search methods can be found in [18]. Our search method can be placed under the Tabu search meta-heuristic in that we allow cost-increasing solutions to direct the search away from local minima, and use a tabu list to prevent from looping back to old solutions. A solution is a vector $w = \{w_1, .., w_n\}$ of weights, with one weight per directed link in the network. We have a solution space W where each weight can take integer values between 1 and 65535. We generate a neighboring solution $i \in N(w)$ by increasing one weight in the

current solution w to divert traffic from the most utilised link (s,t) or change weights to create paths with the same cost to get ECMP routing of traffic over several links from s. We use a l-balanced cost function (as described in the previous section) calculated for the given topology, traffic matrix and required utilisation level l. The cost $f(w)$ for a given weight vector is determined by calculating the shortest paths routing with these weights using Dijkstra's algorithm, adding the traffic matrix, and applying the cost function to the resulting link loads. The starting point is to set all weights to the same value, for instance $w_i = 10$. The search terminates either when we find a solution with utilisation under the threshold l or it stops after a fixed number of iterations.

At the core of our search method is a simple descent search [18] where we:

1. choose an initial weight vector $i \in W$
2. find the neighbor $j \in N(i)$ with lowest cost i.e. $f(j) <= f(k)$ for any $k \in N(i)$.
3. If $f(j) >= f(i)$ then stop. Else set $i = j$ and go to step 2.

This type of search may stop at a local minimum. We therefore allow the search to continue by doing new descents starting from weight sets with higher cost. We use information that becomes available during the search to build a candidate list of weight sets that are used as starting points, and a tabu list of weight sets are used to avoid cycling.

We start by setting all weights to the same value. This gives the shortest paths in number of hops which probably is a good starting point for most real networks; if the link capacities are uniform and the network was built with OSPF/IS-IS routing in mind. Given the network topology, traffic matrix and initial weights, we calculate the ECMP shortest paths, add the traffic matrix, and find the most loaded link (s,t) in the network. If the utilisation is less than *l* then we are done. We have a routing that takes the shortest paths possible and makes sure that no link is utilised to more than the limit *l*. If the link is utilised to more than *l* we start searching for a better weight setting using two strategies:

- the first search strategy is to increase the weight on the overloaded link in controlled steps so to divert more and more demands (or part of demands) from the link. See details in 10.4.3.
- the second search strategy is to find weights to get ECMP routing from s for the demands over (s,t), and so balance the traffic over the outgoing links from s. See details in 10.4.4.

In each iteration of a descent we have a number of neighbor weight settings that we evaluate (one for each weight step and ECMP set described above). If a neighbor weight setting gives a lower cost than the current best (in this iteration) it is saved and used as the starting point in the next iteration. If a candidate weight setting gives a routing with a higher cost than the current best but with a different link than (s,t) as most utilised, then that weight-setting is saved in the candidate list and used as a starting point for another descent search later on.

10.4.3 How to determine weight increments for a link?

If a link (s,t) is over-utilised we want to increase the weight on the link in controlled steps so to divert more and more traffic demands from the link.

To decide the steps in which to increase the weight on (s,t) we first determine the current total weight-cost for each demand routed over (s,t). We then temporarily take away the link (s,t) from our representation of the topology and calculate a new shortest-path routing. For all demands that before were routed over (s,t) we then check how much the weight cost have increased and use this for determining the steps with which to increase the weight on (s,t).

In the example in Figure 10.2, we assume that the two demands $D(1,2)$ and $D(4,2)$ overload the link $(1,2)$. We thus want to divert traffic from the link $(1,2)$ by increasing the weight $w(1,2)$.

We start by determining the increase steps in which to increase the weight $w(1,2)$:

The total weight costs for $D(1,2)$ and $D(4,2)$ are 10 and 40, respectively. If we take away the link $(1,2)$, we get total weight costs of 30 and 50, an increase by 20 and 10 units respectively. From this we decide on the increase steps 10, 15 (mid-point between 10 and 20), 20 and 21 units. We add this to the original $w(1,2) = 10$ and get the candidate weights $w(1,2)$= 20, 25, 30 and 31 to evaluate.

With the first increment $w(1,2) = 20$ we divert half of demand $D(4,2)$ by ECMP while the other half of $D(4,2)$ and all of demand $D(1,2)$ is still routed on $(1,2)$. The next increment $w(1,2) = 25$ diverts all of $D(4,2)$ but keeps all of $D(1,2)$. With $w(1,2) = 30$ we also route half of $D(1,2)$ on another path and with $w(1,2) = 31$ we divert all traffic from $(1,2)$.

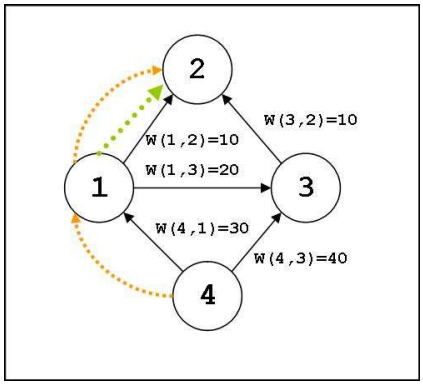


Figure 10.2: Example with an overloaded link (1,2) where traffic can be diverted to other paths by increasing the weight on (1,2) in controlled steps w(1,2)= 20, 25, 30 and 31. With the first increment w(1,2)=20 we divert half of demand D(4,2) by ECMP. The next increment w(1,2)=25 diverts all of D(4,2), and with w(1,2)=30 we route also half of D(1,2) on another path. Finally, w(1,2)=31 diverts all traffic from (1,2).

10.4.4 How to determine ECMP weight settings?

If we have a weight set that results in an overloaded link (s,t) then we want to also evaluate neighbor weight settings where we split traffic demands evenly over the outgoing links from s using ECMP. In order to split a traffic demand ECMP the total weight for each path from s to the demand destination d need to be the same.

Consider, as in Figure 10.3, a node s, the next hops t_i, and the shortest path P_i from each t_i to the destination d. Also consider the corresponding weights $w(s,t_i)$ and total weight cost $w(P_i)$ for a path P_i from t_i to d. One way to achieve ECMP weights is to adjust the weights $w(s,t_i)$ on the outgoing links from s such that:

$$w(s,t_i) = 1 + max_{j=1,..,n}\{w(P_j)\} - w(P_i)$$

This gives the same total cost for each path from s to d.

A possible extension to this is to not always spread the traffic over all possible links but also evaluate different subsets of ECMP weights setting with varying number of outgoing links from s.

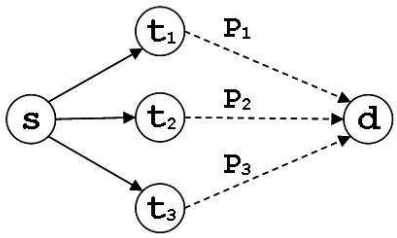


Figure 10.3: Determining ECMP weights

10.4.5 Increment weight on a less utilised link in a path

With high traffic load in the network, link weights can become sensitive to change after some iterations in the search. For instance if we on an overloaded link already have adjusted the weight to split a large demand with ECMP then we can not easily increase the link weight to divert yet another flow without disturbing the existing load balancing.

In order to divert traffic demands to other paths but without disturbing existing splits on the most utilised link we extend the neighborhood in the search. We evaluate weight sets where we instead of changing the weight on the overloaded link (s, t) increment the link weight some step away closer to the demand destination.

10.5 Evaluation

10.5.1 Method

In order to evaluate the *l*-balanced routing and our search method for finding *l*-balanced weights we use real network topologies and traffic matrix data that we scale up to get high loads in the networks. First in Section 10.5.2 we evaluate that the search method works well for finding *l*-balanced weight setting in these scenarios och compare the resulting network loads with optimal *l*-balanced routing and routing with other traffic engineering objectives. The main objective of *l*-balanced routing is to give spare capacity to handle traffic changes. In Section 10.5.2 we investigate how different weight settings handle hotspots where one traffic matrix entry increases.

For the evaluation we use three different data sets that include network

topologies and traffic matrix data from the Geant network [19], and from the European and American sub-networks of a global IP network.

- Network I: the Geant network with 23 nodes, 74 links and 506 demands.
- Network II: the American network with 24 nodes, 110 links and 552 demands.
- Network III: the European network with 11 nodes, 38 links and 110 demands.

The details of the global IP-network, the subnetwork topologies and traffic demands, are described in [20]. For the Geant network we set all link capacities to 10 Gb and scaled up the traffic data to create high loads in the network.

10.5.2 Static scenario: Evaluating the search method

The evaluation shows that the *l*-balanced objective and our search method for finding l-balanced weight settings work well. Figure 10.4 shows comparisons of optimal and weight-based *l*-balanced routing (with *l*=80%) for increasing levels of traffic demand in the Geant network (Network I) and the American network (Network II). The *l*-balanced routing sends the traffic on the shortest paths as long as the utilisation is low in the network. The shape of the curves shows that when we scale up the traffic demand the *l*-balanced method tries to keep the utilisation under *l=0.8*. The figures also show that the weight-based routing is close to the optimal routing which validates that our search method for setting the weights works well. Note that optimal routing minimises the total cost when the *l*-balanced cost function is applied to the utilisation of each link in the network. The utilisation for an individual link (and so the maximum link utilisation) can be higher in the optimal solution if it finds a shorter path that still keeps the utilisation below *l*.

Figure 10.5 shows a comparison between the *l*-balanced routing and other traffic engineering objectives. The minimum-hop routing (with all weights set to 1) , where no attempt is done to adapt the weight setting to the current traffic demand, quickly leads to overload in the network when the traffic demands are increased. The *l*-balanced method sends the traffic on the shortest paths as long as the utilisation is less than the chosen value *l*=0.8. With a low utilisation of the network there is no reason to split the traffic over several paths. The FT cost function used in [2], pushes down the maximum link utilisation already at lower traffic levels. This piece-wise linear cost function consists of several segments which is reflected in the shape of the curve with plateaus where the

maximum link utilisation is pushed down. With minmax routing the objective is to minimise the maximum link utilisation in the network. This routing always balance the load over the network to keep the highest link utilisation down to a minimum. The optimal minmax routing gives a lower bound on how much it is possible to keep down the maximum link utilisation.

10.5.3 Dynamic scenario: Evaluation of robustness

The main purpose with *l*-balanced routing is to give a controlled traffic level and spare capacity to handle uncertainties and sudden changes in the traffic situation. To confirm that the *l*-balanced weight settings fulfil this, we added hotspot traffic (in a magnitude that the *l*-balanced routing should be able to handle) and investigated the resulting link utilisations. Figure 10.6 shows the maximum link utilisations for minimum hop routing, *l*-balanced and FT weight-settings under assumed hotspot traffic in the Geant network scenario. After determining the weights and the routing for a given traffic matrix each of the 506 demands was increased one at a time by 20% of the link capacity. The minimum hop routing gives overloaded links for many of the hotspots at this traffic demand level while the *l*-balanced and FT routing manage to handle the traffic increase without exceeding the capacity of any link. In Figure 10.7 we increase the traffic level. Now the FT routing sometimes results in overloaded links when the hotspot traffic is added. The *l*-balanced routing (with *l*=0.8) on the other hand gives 20% spare capacity and so handle the increase for any of the demands. The minimum hop routing without any traffic engineering gives link overload even without adding hotspot traffic at this demand level and is not shown in the figure.

10.6 Conclusions

L-balanced weight settings give the operator possibility to apply simple rules of thumb for controlling the maximum link utilisation and control the amount of spare capacity needed to handle sudden traffic variations. It gives more controlled traffic levels than other cost functions and more efficient routing for low traffic loads when there is no need to spread traffic over longer paths. In this paper we have presented a heuristic search method for finding *l*-balanced weight settings and shown that the search and the resulting weight settings work well in real network scenarios.

Acknowledgment

The authors would like to thank Anders Gunnar for providing the traffic data used in the evaluation. We also would like to thank Adam Dunkels and Bengt Ahlgren for inspiring and helpful discussions.

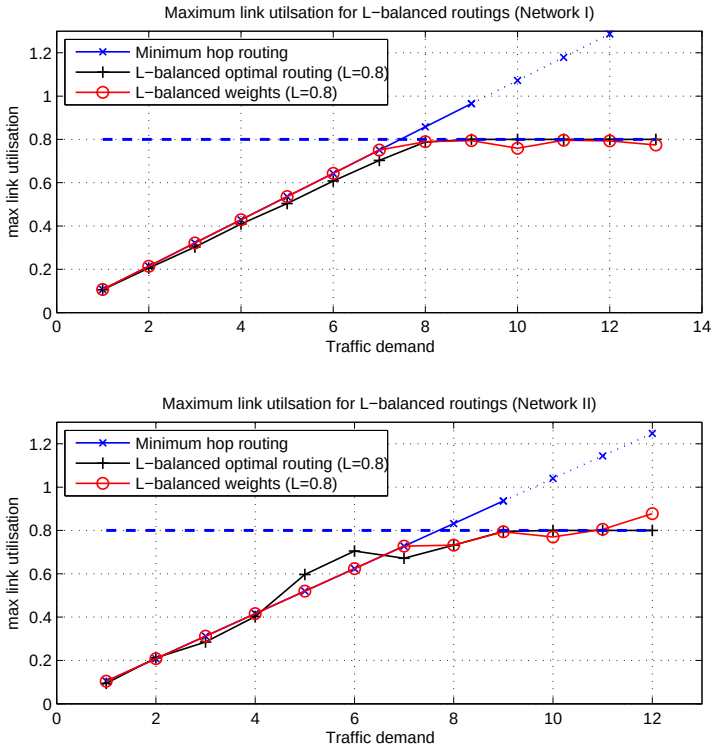


Figure 10.4: Comparison of maximum link utilisations for optimal and weight-based L-balanced routing for different scaled traffic demands in the Geant network (top) and the American network (bottom). The utilisation is kept under the chosen limit l and the weights found by the search heuristic gives a routing close to optimal.

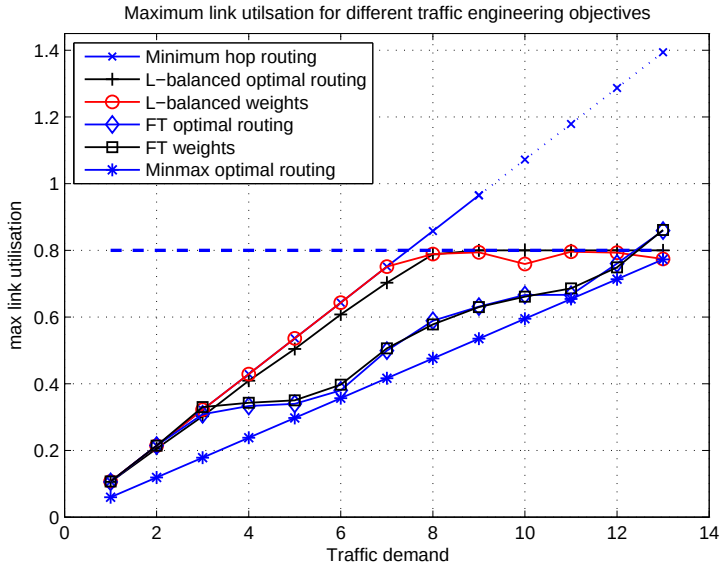


Figure 10.5: Comparison of maximum link utilisations for different traffic engineering objectives in the Geant network.

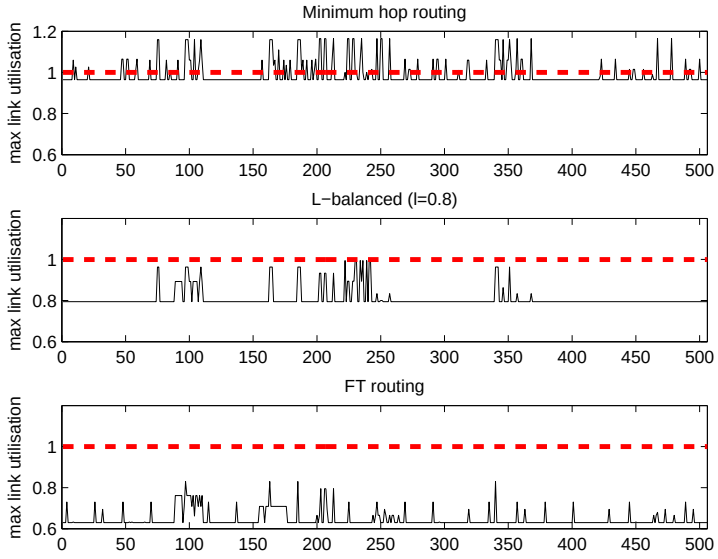


Figure 10.6: Hotspot traffic scenario in the Geant network. Comparison of maximum link utilisations for three weight setting strategies. Minimum hop routing exceeds the link capacity while *l*-balanced and FT routing can avoid overload.

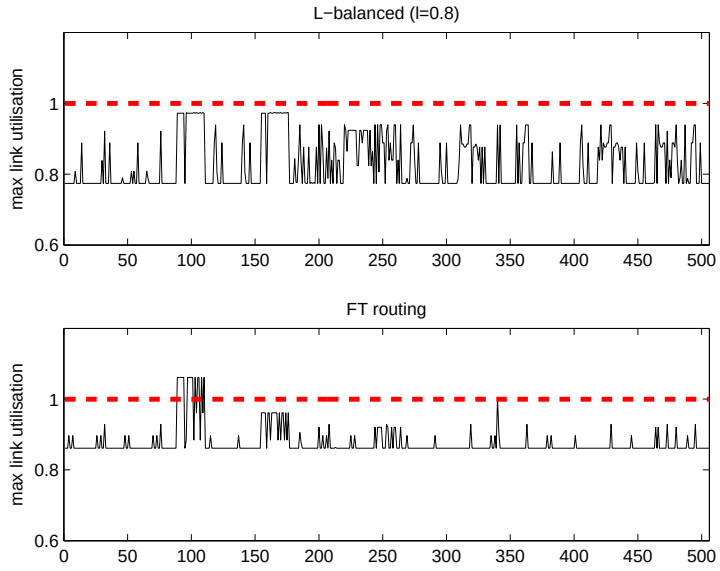


Figure 10.7: Hotspot traffic in the Geant network with scaled up traffic demands. FT routing here gives overloaded links while the L-balanced routing can handle the traffic increase.

Bibliography

[1] H. Abrahamsson, J. Alonso, B. Ahlgren, A. Andersson, and P. Kreuger. A multi path routing algorithm for IP networks based on flow optimisation. In *Proceedings of the Third International Workshop on Quality of Future Internet Services (QoFIS)*, Zürich, Switzerland, October 2002.

[2] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *Proceedings IEEE INFOCOM 2000*, pages 519–528, Israel, March 2000.

[3] M. Pióro and D. Medhi. *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann, 2004.

[4] J. Rexford. Route optimization in IP networks. In Mauricio G.C. Resende and Panos M. Pardalos, editors, *Handbook of Optimization in Telecommunications*. Springer Science+Business Media, 2006.

[5] B. Fortz and M. Thorup. Optimizing OSPF/IS-IS weights in a changing world. *IEEE Journal on Selected Areas in Communications*, 20(4):756–767, May 2002.

[6] K.G. Ramakrishnan and M.A. Rodrigues. Optimal routing in shortest path data networks. *Lucent Bell Labs Technical Journal*, 6(1), 2001.

[7] A. Gunnar, H. Abrahamsson, and M. Söderqvist. Performance of Traffic Engineering in Operational IP-Networks: An Experimental Study. In *Proceedings of 5th IEEE International Workshop on IP Operations and Management*, Barcelona, Spain, October 2005.

[8] A. Sridharan, R. Guérin, and C. Diot. Achieving Near-Optimal Traffic Engineering Solutions for Current OSPF/IS-IS Networks. In *IEEE Infocom*, San Francisco, March 2003.

[9] David Applegate and Edith Cohen. Making Intra-Domain Routing Robust to Changing and Uncertain Traffic Demands: Understanding Fundamental Tradeoffs. In *Proceedings of ACM SIGCOMM*, Karlsruhe, Germany, August 2003.

[10] J. Alonso, H. Abrahamsson, B. Ahlgren, A. Andersson, and P. Kreuger. Objective functions for balance in traffic engineering. Technical Report T2002:05, SICS – Swedish Institute of Computer Science, May 2002.

[11] S. Balon, F. Skivee, and G. Leduc. How Well Do Traffic Engineering Objective Functions Meet TE Requirements? In *Proceedings of IFIP International Networking Conference*, Coimbra, Portugal, May 2006.

[12] C. Zhang, Z. Ge, J. Kurose, Y. Liu, and D. Townsley. Optimal Routing with Multiple Traffic Matrices: Tradeoffs between Average Case and Worst Case Performance. In *Proceedings of ICNP 2005*, Boston, USA, November 2005.

[13] C. Zhang, Y. Liu, W. Gong, J. Kurose, R. Moll, and D. Townsley. On Optimal Routing with Multiple Traffic Matrices . In *Proceedings of Infocom 2005*, Miami, USA, March 2005.

[14] H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, and A. Greenberg. Cope: Traffic engineering in dynamic networks. In *Proceedings of ACM SIGCOMM*, Pisa, Italy, September 2006.

[15] Anders Gunnar and Mikael Johansson. Robust routing under bgp reroutes. In *Proceedings of Globecom 2007*, Washington, DC, USA, 2007.

[16] Dahai Xu, Mung Chiang, and Jennifer Rexford. DEFT: Distributed exponentially-weighted flow splitting. In *IEEE Infocom*, Anchorage, Alaska, USA, May 6–12, 2007.

[17] Dahai Xu, Mung Chiang, and Jennifer Rexford. Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering. In *IEEE Infocom*, Phoenix, Arizona, USA, April 2008.

[18] E. Aarts and J. K. Lenstra, editors. *Local search in combinatorial optimization*. Princeton University Press, 2003.

[19] *The Geant network*. http://www.geant.net.

[20] A. Gunnar, M. Johansson, and T. Telkamp. Traffic Matrix Estimation on a Large IP Backbone - a Comparison on Real Data. In *Proceedings of ACM Internet Measurement Conference*, Taormina, Sicily, Italy, October 2004.