Copyright

by

Sangkyu Park

2002

The Dissertation Committee for Sangkyu Park
certifies that this is the approved version of the following dissertation:

Traffic Engineering in Multi-service Networks: Routing, Flow Control and Provisioning Perspectives

Committee:

Gustavo de Veciana, Supervisor

San-qi Li

Edward J. Powers

Scott Nettles

John Hasenbein

Traffic Engineering in Multi-service Networks: Routing, Flow Control and Provisioning Perspectives

by

Sangkyu Park, B.S., M.S.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

December 2002

To my parents

Acknowledgments

This thesis is a result of over six-years of research work that I carried out in the Telecommunications and Information Systems Engineering (TISE) division (later re-organized to become the Wireless Networking and Communications Group (WNCG)) in the ECE department, the University of Texas at Austin. The ideas, models and results contained in this thesis could not have been achieved without the supports of many people.

I wish to convey my deep gratitude to Prof. Gustavo de Veciana who served as my supervisor since the summer, 2001. Majority of the idea and work contained here could have been enlightened, sharpened and organized through his guide. His energy has given an enormous boost to my thesis work. Most of all, I wish to thank him sincerely for his enthusiasm toward research and the great perseverance of waiting while I was in time-consuming progress. Without his support, this thesis could not have existed.

In the beginning of my work I had great help from Dr. San-qi Li when he served as my supervisor when he was with the University of Texas at Austin. I give special thanks to him for the continuing support and encouragement.

I also give my great thanks to the professors who agreed to serve in my dissertation committee. Prof. Scott Nettles, Prof. Edward Powers and Prof. John Hasenbein gave precious feedback on the ideas contained in this thesis.

It has been my pleasure to co-work with my colleagues; Dr. Sang Aimin, Steven Weber, Dr. Na Li, Dr. Yetik Serbest and Dr. Jay S. Yang. They gave invaluable advice and comments on my work.

Last, I give my deep thanks to my parents and my brother for their endurance and support.

Sangkyu Park

The University of Texas at Austin
December 2002

Traffic Engineering in Multi-service Networks: Routing, Flow Control and Provisioning Perspectives

Publication No. ____________________

Sangkyu Park, Ph.D.
The University of Texas at Austin, 2002

Supervisor: Gustavo de Veciana

A possible evolution for current telecommunication networks is towards an architecture which is increasingly connection oriented or based on virtual overlays, e.g., MPLS, as a means to support multiple service types and/or customers. At the same time such networks would continue support large amounts of traffic based on best effort service paradigm currently used in TCP/IP networks. This thesis investigates several aspects in the design and operation of such networks.

As technologies continue increasing the capacity of network resources to meet increasing traffic loads the number of users contending for a given congested resource is likely to grow. When such sharing is mediated by a best effort paradigm based on TCP, one might ask whether the effectiveness of the underlying mechanisms will scale. In this thesis we begin by assessing the scalability of active queue management combined with flow control mechanisms for higher capacity links and larger

numbers of ongoing flows. We propose and evaluate a simple algorithm for congestion avoidance which doesn't require that the number of connections be known (or estimated) but exhibits robust performance even when the number of contending flows varies over a wide range. This provides a promising avenue for achieving good performance over a wide range of systems.

The ability to perform a flexible routing of traffic loads across network resources is a key to enable providers to support a wide range of typically time varying loads. Nevertheless due to possible instabilities and unpredictability, operators have for the most part been unwilling to use dynamic routing mechanisms that adapt to changing loads on the fly. Instead they have opted to assign *fixed* administrative weights to links, and have traffic routed based on simple shortest path routing algorithms. In this thesis we consider how to set those weights given a priori information on the traffic loads. This corresponds to solving an inverse problem: determine a set of fixed weights for network links such that shortest path routing of traffic gives a good flow distribution across the network. When the traffic being routed includes connection oriented services, a good distribution of load over the network, must consider both bandwidth and connection processing and maintenance resources. Indeed connection oriented services typically require network resources to process signaling messages for setup and tear down, as well as periodic signaling for maintenance purposes. We propose and evaluate an approach to solving this inverse problem when the objective is to balance loads across both bandwidth and connection signaling resources in the network.

One way providers can deal with scalability issues associated with supporting multiple service classes or customer types is to setup overlay networks over a

common infrastructure. The last question we address in this thesis is associated with routing and provisioning virtual private networks (VPNs) when only a limited characterization of the traffic demands is available or desirable. Specifically we focus on the 'hose model' where only an aggregate characterization/constraints on traffic entering and leaving a collection of endpoints is available. To this end we devise an approach to routing and provisioning VPNs so as to exploit spatio-temporal multiplexing gains to minimize provisioning cost. We study the same questions when the provider's objective is to maximize the lifetime of the routing decisions and/or capacity to carry new VPN requests rather than minimizing per VPN costs.

Table of Contents

Acknowledgments **v**

Abstract **vii**

List of Tables **xiv**

List of Figures **xv**

Chapter 1 Introduction **1**

Chapter 2 Threshold-based binary feedback for congestion avoidance **4**

2.1 Introduction 4

2.2 Throughput variance analysis 8

2.3 Threshold-based marking for explicit binary feedback 12

2.4 Simulation results and discussion 18

2.5 Conclusions 26

Chapter 3 Traffic engineering minimizing congestion of both traffic load and signaling loads **27**

3.1 Introduction 27

3.2 Minimum congestion routing 34

3.2.1 Congestion 34

3.2.2 Multiple resources 37

3.3 Modelling and algorithm 38

3.3.1 Minimum congestion optimization problem 40

3.3.2 Derived problem with tight constraint 42

3.3.3 Dual problem formulation for the derived problem with tight constraint 44

3.3.4 Heuristic dual problem for the derived problem with tight constraint 46

3.4 Numerical experiments 49

3.4.1 Topology and test scenario 49

3.4.2 Results and discussion 50

3.5 Conclusions 55

Chapter 4 Routing and provisioning VPNs on hose traffic models and/or constraints 57

4.1 Introduction 57

4.1.1 Contributions of this chapter 62

4.1.2 Summary of related work 64

4.1.3 Organization of chapter 65

4.2 Hose traffic descriptors and link provisioning functions for VPN trees 66

4.2.1 Call level QoS - voice application 69

4.2.2 Packet level QoS - bursty data traffic application 71

4.2.3 Call and packet level QoS - voice application with silence suppression 73
4.3 Routing VPN trees - minimizing provisioning cost 74
4.3.1 Problem formulation 74
4.3.2 Problem characteristics and algorithm 76
4.3.3 Heuristic algorithm 83
4.3.4 Results 86
4.4 Routing VPN trees - enhancing network lifetime 90
4.4.1 Problem formulation 90
4.4.2 Heuristic algorithm 92
4.4.3 Network growth model 94
4.4.4 Results 97
4.5 Further issues 103
4.5.1 Extending the hose service model - cutset and multi-class constraints 103
4.5.2 Hose based VPN service model - a customer's perspective . . 104

Chapter 5 Conclusion 108

Appendix A Addendum to Chapter 2 111

A.1 Linear program formulation for the piecewise linear congestion cost model 111

Appendix B Addendum to Chapter 4 114

B.1 Minimizing provisioning cost routing experiment results - Pure Random model topology 114

Bibliography **120**

Vita **128**

List of Tables

2.1 Variance of total arrival rate. 13
2.2 Bottleneck link performance (N4-N5). 19

3.1 The notation table. 38
3.2 Link metric setting algorithm. 48
3.3 The offered load table. 50

4.1 Notation table for VPN hose model. 67

List of Figures

2.1 Simple TCP flow model. 9

2.2 Rate allocation(two source (x1 and x2) case: RED). 11

2.3 TMCA algorithm - marking. 13

2.4 Modified TCP slowdown for TMCA (β=1/8). 14

2.5 Rate allocation(two source (x1 and x2) case: TMCA). 15

2.6 Modified TCP slowdown for TMCAr (β=1/8). 17

2.7 Topology for simulation. 18

2.8 RED queue size (max_p=0.1). 21

2.9 TMCA queue size (q^*=60). 21

2.10 Utilization. 22

2.11 Packet loss ratio. 22

2.12 Utilization. 23

2.13 Packet loss ratio. 23

2.14 Packet loss ratio (TMCAr). 24

2.15 Utilization (TMCAr). 25

2.16 Queue size average (TMCAr). 25

3.1 $\Phi(l, c)$: congestion cost function [Fortz et al.]. 34

3.2 The topology with 15 nodes [Kodialam et al.]. 49

3.3 Total congestion cost. 51

3.4 Max. utilization (Bw). 52

3.5 Max. utilization (Cn). 52

3.6 Resource consumption (Bw). 53

3.7 Resource consumption (Cn). 53

4.1 VPN traffic characterization. 58

4.2 VPN routing structures. 66

4.3 Provisioning shared trees. 67

4.4 Provisioning cost - voice application. 71

4.5 Provisioning cost - bursty data traffic. 72

4.6 VPN shared tree routing problem. 74

4.7 Routing shared trees - algorithm. 83

4.8 Routing shared trees - intuition. 84

4.9 Heuristic algorithm to solve NIP(G,v). 86

4.10 Algorithm to route near-optimal tree (symmetric case). 87

4.11 Results versus number of endpoints with Waxman model topology. . 88

4.12 Results versus average node degree with Waxman model topology. . 89

4.13 Congestion sensitive routing. 91

4.14 Congestion sensitive routing - future growth. 92

4.15 Heuristic algorithm to solve NIP(G,v) for load balancing. 93

4.16 NSF topology. 97

4.17 Results versus parameter β with real network topology (NSF net). . 98

4.18 Results versus parameter β with real network topology (NSF net) - continued. 99
4.19 Confidence interval of percentage increase in average lifetime - Scenario 1 (NSF net). 99
4.20 Results versus parameter β with Waxman model topology. 100
4.21 Results versus parameter β with Waxman model topology - continued. 101
4.22 Confidence interval of percentage increase in average lifetime - Scenario 1 with Waxman model. 102
4.23 Extending the hose service model. 103
4.24 Large file to share with all endpoints. 105
4.25 Application-level multicasting - load balancing 105

B.1 Results versus number of endpoints with Pure Random model topology.115
B.2 Results versus average node degree with Pure Random model topology.116
B.3 Results versus parameter β with Pure Random model topology. . . . 118
B.4 Confidence interval of percentage increase in average lifetime - Scenario 1 with Pure Random model. 119

Chapter 1

Introduction

A possible evolution for current telecommunication networks is towards an architecture which is increasingly connection oriented or based on virtual overlays, e.g., MPLS, as a means to support multiple service types and/or customers. At the same time such networks would continue support large amounts of traffic based on best effort service paradigm currently used in TCP/IP networks. This thesis investigates several aspects in the design and operation of such networks.

As technologies continue increasing the capacity of network resources to meet increasing traffic loads the number of users contending for a given congested resource is likely to grow. When such sharing is mediated by a best effort paradigm based on TCP, one might ask whether the effectiveness of the underlying mechanisms will scale. In this thesis we begin by assessing the scalability of active queue management combined with flow control mechanisms for higher capacity links and larger numbers of ongoing flows. We propose and evaluate a simple algorithm for congestion avoidance which does not require that the number of connections be known (or estimated) but exhibits robust performance even when the number of contending flows varies over a wide range. This provides a promising avenue for achieving good performance over a wide range of systems.

The ability to perform a flexible routing of traffic loads across network resources is a key to enable providers to support a wide range of typically time varying loads. Nevertheless due to possible instabilities and unpredictability, operators have for the most part been unwilling to use dynamic routing mechanisms that adapt to changing loads on the fly. Instead they have opted to assign *fixed* administrative weights to links, and have traffic routed based on simple shortest path routing algorithms. In this thesis we consider how to set those weights given a priori information on the traffic loads. This corresponds to solving an inverse problem: determine a set of fixed weights for network links such that shortest path routing of traffic gives a good flow distribution across the network. When the traffic being routed includes connection oriented services, a good distribution of load over the network, must consider both bandwidth and connection processing and maintenance resources. Indeed connection oriented services typically require network resources to process signaling messages for setup and tear down, as well as periodic signaling for maintenance purposes. We propose and evaluate an approach to solving this inverse problem when the objective is to balance loads across both bandwidth and connection signaling resources in the network.

One way providers can deal with scalability issues associated with supporting multiple service classes or customer types is to setup overlay networks over a common infrastructure. The last question we address in this thesis is associated with routing and provisioning virtual private networks (VPNs) when only a limited characterization of the traffic demands is available or desirable. Specifically we focus on the ‘hose model’ where only an aggregate characterization/constraints on traffic entering and leaving a collection of endpoints is available. To this end we devise an approach to routing and provisioning VPNs so as to exploit spatio-temporal multiplexing gains to minimize provisioning cost. We study the same questions when the provider’s objective is to maximize the lifetime of the routing decisions and/or capacity to carry new VPN requests rather than minimizing per VPN costs.

Chapter 2

Threshold-based binary feedback for congestion avoidance

2.1 Introduction

Congestion control/avoidance has been studied for more than ten years [1]. The purpose of such mechanisms is to recover from or avoid network collapse when congestion occurs. Closed-loop implicit binary feedback, as used in Transmission Control Protocol (TCP) is an example of the use of congestion control in the Internet. While the type of flow and congestion control used in TCP, and its variants, has been successful, there are still many performance concerns such as estimating bottleneck bandwidths, modelling packet loss events and modelling packet transit delays as indicators of congestion periods [2].

By *implicit* binary feedback we mean an end-to-end protocol at source hosts identifies congestion by detecting lost packets. Because it is not timely and requires retransmission, this detection scheme is not efficient. Thus it is preferable to have more proactive mechanisms to recover from periods of congestion and to achieve a high network resource utilization [3]. This has prompted the Internet Engineering Task Force (IETF) to recommend the use of active queue management [4] in conjunction with explicit congestion notification (ECN) [16].

ECN is classified as an *experimental* protocol by the IETF, which means that it has been specified but may not progress to full standardization until some experience is gained with it. It is an important addition to the Internet Protocol (IP) service model which is geared at supporting Quality of Service (QoS)[5]. ECN requires routers to have some form of active queue management such as Random Early Detection (RED)[4], and requires modifications to the current protocol at both the routers and end hosts.

Active queue management, and ECN form an *explicit* end-to-end binary feedback system. When congestion arises, an active queue management scheme at network routers marks packets so as to notify the source host - the *congestion experienced*(CE) bit in the packet is set. In turn the transport protocol at the receiver sets the *ECN-echo* bit in the acknowledgement(ACK) packets when receiving marked data packets. Finally the transport mechanism at a source host reduces its congestion window when it receives a marked acknowledgement. At the expense of modifying the current protocols, ECN avoids dropping packets unnecessarily for congestion notification, and decouples congestion notification from transmission errors.

At each router, an active queue management scheme starts marking data packets when congestion is detected to reduce total arrival rates before queue overflow occurs. Take RED, for example, where marking is done in a random manner paced on a calculated probability p. Finding the appropriate value for p under various scenarios is not an easy task. To enhance the performance at the routers, many ideas have been suggested, such as Stabilized-RED (SRED)[6], Self-Configuring RED (SCRED)[14], BLUE[7], Rate Based n-RED[8] and MARS[12]. Yet, there is no final conclusion about which algorithm is good enough to be implemented.

Another approach to indicate congestion is to feedback rate information. Rate-based feedback was introduced in Asynchronous Transfer Mode (ATM) network for Available Bit Rate (ABR) control [9][13]. Many approaches were proposed for tracking the feedback information (e.g. a fair share for each flow) without knowing per-flow information at routers [11]. The drawbacks of these approaches are: (1) control packets have to be sent out regularly to carry the rate information, (2) the routers have to periodically calculate the fair share; and (3) flows may become synchronized and cause oscillation by conforming to the same control at almost the same time. It might cause oscillation. Due to these complexities, currently not so many routers support this type of operation.

In this chapter, we propose an alternative to the above approaches. We will use threshold-based marking for congestion avoidance (TMCA) at routers - this is a type of explicit binary feedback. The routers only need to monitor the queue size (instantaneous or average), and mark packets once the queue size exceeds the threshold. This approach imposes a minimal overhead at the routers. [1] In this

---

[1] Many ATM switches already have this capability.

chapter we study the performance of this approach and compare it with traditional active queue management schemes such as RED.

TMCA marks *all* arriving packets once the filtered queue length q_{avg} exceeds a target queue level q^*, and stops marking when queue average goes lower than q^*. When the queue length is in the region, $q_{avg} \leq q^*$, there is no marking, so all flows are allowed to increase their congestion windows. Therefore we can maintain high utilization and thus high throughput. In the region where $q_{avg} > q^*$, all flows are responding to the marking, and queue levels are firmly controlled. Loss ratios can thus be reduced.

Note that TMCA's marking action brings strong correlations among flows. The flows tend to react to marking at the same time, and therefore tend to synchronize. In this respect, it is similar to rate-based feedback. The difference is that TMCA provides only one-bit feedback while rate-based schemes provide multi-bit feedback.

Active queue management, e.g. RED, which uses binary feedback, tends to randomly select flows and sends congestion notifications to only a partial set of flows. The marking probability in active queue management affects the manner in which congestion notifications are fed back to sources. In addition, for a given marking probability, different marking schemes produce different queue variance[15]. In designing a marking scheme, an important thing to take into account is the manner in which rate reduction actions of flows evenly distributed on the time frame, because a well dispersed marking produces less queue variance.

In TMCA, the marking probability is 1.0 upon congestion, i.e. $q_{avg} > q^*$, at which point all incoming packets are marked. The impact of congestion notification depends on the extent to which the notified sources reduce their window size. To make TMCA work properly, the rate cut of each flow should be small. Otherwise, the synchronized cuts would be too large. By reducing the TCP's multiplicative decrease coefficient β from 1/2 to 1/8, we can reduce such excessive reactions. To reduce fluctuations caused by synchronized reactions to markings, we can further introduce random delays to source reactions. In this case we will denote the scheme TMCAr. Note that compared to RED, TMCA has fewer parameters to tune since it only has 0.0 or 1.0 marking probability. In the same way, TMCAr has fewer parameter to tune at queues but it has an additional parameter, the random delay, at the source side to tune.

The remainder of the chapter is organized as follows. Section 2.2 provides analysis of throughput variance. In Section 2.3 the structure of TMCA is described. Simulation results and discussions can be found in Section 2.4. Finally, Section 2.5 concludes the chapter.

2.2 Throughput variance analysis

Suppose an individual TCP flow increases its congestion window until it uses up the available capacity on the connection path, and queues along the path feedback congestion notification in the form of RED markings. Once a TCP source detects congestion, it reduces its congestion window to half of its current size. A flow's window size W and its throughput T (i.e. rate) have the following linear relation, where RTT denotes the Round Trip Time :

$$T = \frac{W}{RTT}. \tag{2.1}$$

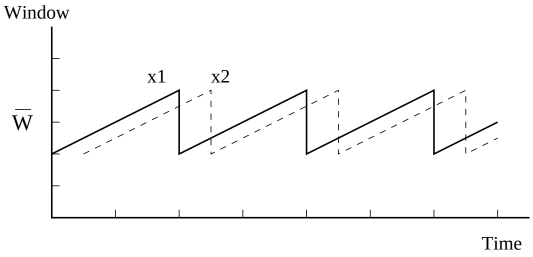


Figure 2.1: Simple TCP flow model.

As is well known RED's marking and TCP's flow control scheme work together to determine the window size. At the same time, the total flow arrival rate to the bottleneck queue depends on the sum of the traversing flows' window sizes. Therefore the characteristics of the total arrival rate directly affect the queue behavior at the bottleneck link. Let us consider the packet loss ratio e and utilization ρ at the bottleneck router as performance metrics. The goal is to achieve small e as well as large ρ. For better performance, it is preferable that the mean arrival rate is maintained at a desired level so that the queue is neither under-utilized nor overloaded. Therefore the queue's occupancy level is a good indication of whether the arrival rate is controlled at a proper magnitude in comparison with the service rate.

Another desirable property is that the variance of arrival rate should be small. With small variance, we can save queue space and reduce jitter, while large variance of arrival rate may incur both under-utilization and high packet loss ratio. In [15], the queue variance for ECN was studied with an ideal sawtooth shaped window size sample path associated with TCP Reno[18], as plotted in Fig. 2.1, where two arbitrary TCP flows, x1 and x2, are shown. Each flow increases its rate until it receives feedback in the form of a binary congestion notification, and reduces its window by half. We follow this approach to design our algorithm, and consider the window size W as a random process and setup a simple model to analyze the queue variance. We assume W is uniformly distributed in $[\frac{2}{3}\overline{W},\frac{4}{3}\overline{W}]$ where $\overline{W}$ is the mean of W.

Assume that M flows share the same Round Trip Times (RTTs) and pass through a single bottleneck router, with capacity C. Thus each flow's throughput is approximately $\frac{C}{M}$. In Fig. 2.1, the variance of each window W is $\frac{\overline{W}^2}{27}$. The variance of each flow's arrival rate can be shown to be

$$VAR(T) = \frac{C^2}{27M^2}. \tag{2.2}$$

As in [15], if we assume all the flows are out of phase, i.e. all the flow's phases are evenly spaced, the variance of the total arrival rate is still $\frac{C^2}{27M^2}$. If all the flows are in phase, the variance of the aggregated window size is $\frac{1}{27}(M\overline{W})^2$ which is M times the variance of an individual flow size and the variance of the total arrival rate becomes $\frac{C^2}{27}$. If the phases of all flows are identically and independently distributed variables, the variance of the total arrival rate is the sum of the individual variances and given by $\frac{C^2}{27M}$.

As can be seen there are three factors that affect the arrival rate characteristics at a bottleneck router. First, mean and variance of individual flow's rate directly affects the total arrival rate. Second, the phases of each flow's sawtooth-shaped rate, i.e. if all the flows reduce their rate upon congestion (for instance, if x1 and x2 in Fig. 2.1 are in phase), this may cause under-utilization of the queue. This problem is known as the global synchronization[10]. Third, the effectiveness of a packet marking scheme depends on the number of flows. As we can see in Equation (2.2), when there are fewer number of flows, the variance is larger. Because of its

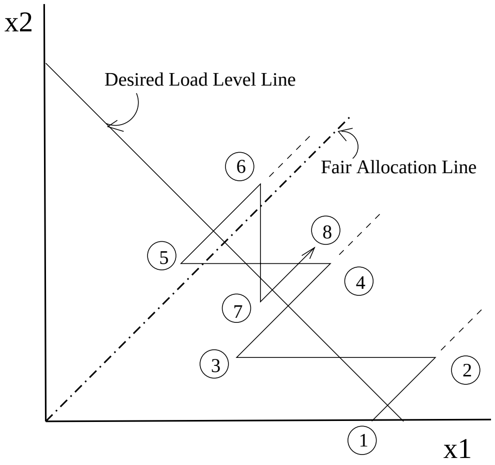


Figure 2.2: Rate allocation(two source (x1 and x2) case: RED).

fixed maximum marking probability, RED shows limited performance in covering varying number of flows[14].

The rate allocation shown in Fig. 2.2 illustrates the congestion avoidance of RED[1] in a two-source case. The optimal allocation is the intersection point of the desired load level line and the fair allocation line. When n is the number of flows and x_i denotes the allocated rate for a flow i in $[1, n]$, fairness can be defined [1] as

$$F(x) = \frac{(\sum_{i=1}^{n} x_i)^2}{n(\sum_{i=1}^{n} {x_i}^2)}. \tag{2.3}$$

The arrow line which traverses the points from (1) to (8) indicates a rate adaptation scenario. Assume at first there is only one flow x1, and it has all the resources. The rate is oscillating on the x1 axis crossing back and forth the desired load level line. When a new connection for flow x2 is set up, additive rate increases in both flows cause the rate allocation (x1,x2) to move to point (2). The path's slope is 1 and is the same as the fair allocation line. Similar increase actions happen from point (3) to (4), (5) to (6) and (7) to (8). RED marks packets of either x1 or x2 with a certain probability. For flow x1, rate reduce to half at point (2) and point (4). For x1, the rate reduces at point (6). Note that even though less likely, x2 reduces its rate at point (2) and (4), and that x1 reduces at point (6). When and if this happens, the fairness gets worse since the point moves away from the fair allocation line.

In Fig. 2.2, the desired load level line is an imaginary line which passes the desired final fair average rate allocation level in steady state. In RED there is no specific parameter to capture the desired allocation. Note that in RED, rate reduction can happen even when the total allocation is below the desired load level line given any marking threshold and probability setting. In other words, the marking probability may not be zero when the total arrival rate is below the desired level, where rate increases are actually preferable. In addition, the figure shows the binary rate cut causes big fluctuations in the total rate allocation, i.e. x1+x2. In the next section we will try to reduce the queue variance caused by these factors.

2.3 Threshold-based marking for explicit binary feedback

In TMCA, we mark all packets once queue level exceeds a queue threshold $q*$. Once the queue level goes below this threshold, we stop marking packets all together.

As pointed out in the previous section, three factors affect the arrival rate characteristics. We are interested in reducing rate variance to enhance queue performance as follows. First, by reducing the ratio of rate reduction, we may reduce mean and variance of each flow. For example, we may modify TCP by setting its binary rate reduction upon congestion feedback to one eighth of the current flow's

```
For each packet arrival
    Calculate the average queue size $q_{avg}$;
    if $q_{avg} \geq q^*$
        mark all arriving packet;
```

Figure 2.3: TMCA algorithm - marking.

rate.

Let us estimate the total rate variance based on the ideal saw-tooth shaped model as in the previous section, let W' be a new random variable for the window size and let $\overline{W'}$ be its mean. Then W' is uniformly distributed over $[\frac{14}{15}\overline{W'}, \frac{16}{15}\overline{W'}]$. The minimum value of W' is $\frac{7}{8}$ of the maximum value. Again, each flow's throughput T' is approximately $\frac{C}{M}$ and the variance of each window W' is now $\frac{\overline{W'}^2}{675}$. We can get the variance of individual flow rates T' as

$$VAR(T') = \frac{C^2}{675M^2}. \tag{2.4}$$

The variance of total rate under various conditions and the results are shown in Table 2.1. The case when the coefficient β of multiplicative decrease equals $\frac{1}{8}$ shows much smaller total rate variance than the original case where β equals $\frac{1}{2}$. As long as $\overline{W'}$ does not exceed 5 times of $\overline{W}$, which is a very rare case, TMCA's variance is smaller. This simple model illustrates how TMCA effectively reduces the rate variance.

Table 2.1: Variance of total arrival rate.

Rate Decrease	In-Phase	Random Phase	Out-Of-Phase
$\beta = \frac{1}{2}$	$\frac{C^2}{27}$	$\frac{C^2}{27M}$	$\frac{C^2}{27M^2}$
$\beta = \frac{1}{8}$	$\frac{C^2}{675}$	$\frac{C^2}{675M}$	$\frac{C^2}{675M^2}$

```
$\beta$ : multiplicative decrease coefficient
cwnd : congestion window size of TCP
TimerRTT : a timer that expires on every RTT

On every ACK

    if ( (timerRTT == 0) and (ACK==ECN) )
    {     cwnd = cwnd * ( 1 - $\beta$ );
          timerRTT = cwnd;
    }
    else    cwnd = cwnd + 1/cwnd ;

    if (−−timerRTT < 0 ) timerRTT = 0;
```

Figure 2.4: Modified TCP slowdown for TMCA (β=1/8).

Fig. 2.3 shows the dynamics of our simple TMCA marking algorithm. The units of q_{avg} and q^* are in packets. The average queue length is calculated using an exponentially weighted moving average filter, i.e.,

$$q_{avg}(t+1) = (1 - w_q)q_{avg}(t) + w_q q(t+1). \tag{2.5}$$

The parameter $w_q \in (0, 1)$ determines the cutoff of the filter, and $q(t)$ denotes the instantaneous queue size. The reason to introduce filtering is to avoid excessive marking or congestion indications upon slight bursts of packet arrivals.

In the proposed scheme sources reduce their congestion window by $\frac{1}{8}$ when they receive an acknowledgment packet marked with an ECN-echo. Moreover until congestion window ('cwnd') number of ACK's are received (i.e. one RTT time elapses), there are no further reductions in window size. Meanwhile, the congestion window size is increased by 1 during each RTT time. Fig. 2.4 briefly illustrates the algorithm, where the unit of congestion window 'cwnd' is in packets.

Note that the modest reduction of β used in TMCA is necessary since all

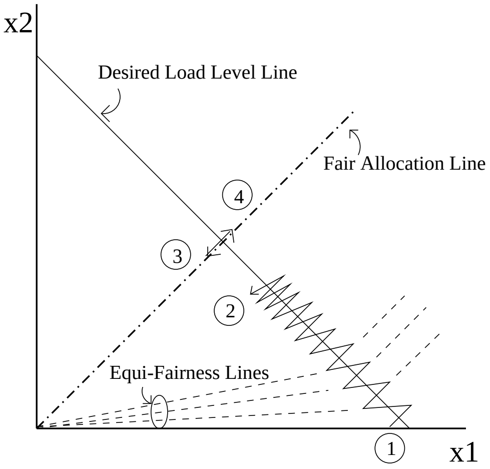


Figure 2.5: Rate allocation(two source (x1 and x2) case: TMCA).

the incoming packets will be marked upon congestion. The original 1/2 reduction, if used with TMCA marking, causes over-reaction to the congestion indications and therefore under-utilization of resources. The additive increase of the window size is 1 packet for every RTT. The additive increase combined with multiplicative decrease guarantees the fair bandwidth sharing among flows [1].

Fig. 2.5 illustrates the two-source case of the rate allocation under our TMCA scheme. Note that in TMCA all packets are marked at bottleneck router and a TCP source receiving ECN react by reducing its rate. The rate reduction is multiplicative decrease which follows equi-fairness line toward the origin in the rate allocation plot. This multiplicative rate reduction preserves the fairness value defined in (2.3). When there is no further congestion feedback, all flows additively increase following the line of slope 1 in the plot. The additive increase line is parallel to the fair allocation line. Thus fairness will be improved during this transition.

The arrowed line from point (1) to (2) in Fig. 2.5 shows one typical scenario of a rate transition. It starts from a state where all the capacity is allocated to source x1 before source x2 is set up. We can see rate reduction with $\beta = \frac{1}{8}$ produces fewer oscillations, and therefore reduces the variance of the total rate x1+x2. Marking with a probability of 1.0 when congestion arises moves the rate towards the fair allocation consistently. Eventually it reaches the fair allocation line and the total rate x1+x2 oscillates between points (3) and (4).

We now consider a further improvement. In the previous section, we saw that the phase of each flow's rate reduction affects the variance of total rate. To reduce total rate's fluctuation, we introduce a randomized delay in TCP's reaction to the marked packets. The key idea is to disperse the rate reduction action over time so as to make the aggregate rate more even. The pseudo code in Fig. 2.6 outlines the algorithm. Since TMCA is now accompanied by TCP with randomized delays, we call this new algorithm as TMCAr.

The modified TCP for TMCAr has three states. Once it gets a congestion indication, it generates a random delay uniformly distributed in the range [0,RTT/2] and changes its state(TCPstate) to 1. In state 1, congestion window size is kept unchanged for the amount of the delay. Then, it reduces the window size and changes to state 2. TCP stays at state 2 by reducing rates by $\frac{1}{8}$ every RTT as long as there is a congestion indication. Otherwise, when RTT timer expires, the state changes back to 0.

In the next section, we will explore the performance of TMCA and TMCAr, and compare it with RED for various parameter settings.

β : multiplicative decrease coefficient
cwnd : congestion window size of TCP
TimerRTT : a timer that expires on every RTT
TCPstate : a state variable to control delayed decrease

On every ACK

```
if ((TCPstate == 0) and (timerRTT == 0) and (ACK == ECN))
{     u = randomUniform();
      Delay = u * cwnd * 0.5; /* random delayed rate reduction */
      RateReductionTime = now + Delay;
      TCPstate = 1;
}
else if ( (TCPstate == 1) and (RateReductionTime <= now))
{     cwnd = cwnd * ( 1 - $\beta$ ); /* rate reduction */
      timerRTT = cwnd;
      TCPstate = 2;
}
else if (TCPstate == 2)
{
     if ( (ACK == ECN) and (timerRTT == 0) )
     {     cwnd = cwnd * ( 1 - $\beta$ ); /* rate reduction */
           timerRTT = cwnd;
     }
     else if (timerRTT == 0)
     {     TCPstate = 0;
     }
}
if (--timerRTT < 0 ) timerRTT = 0;

 if (TCPstate ! = 1)  cwnd = cwnd + 1/cwnd ; /* linear rate increase */
```

Figure 2.6: Modified TCP slowdown for TMCAr (β=1/8).

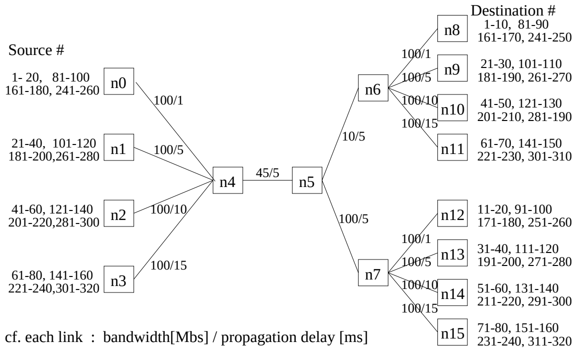


Figure 2.7: Topology for simulation.

2.4 Simulation results and discussion

To explore the effectiveness of the proposed approach, we ran simulations using network simulator NS version 2 [17] and we compared the performance with RED [10]. A variety of scenarios were investigated, including single and multiple bottleneck links, various RTTs, and dynamically changing numbers of flows. The simulation topology is shown in Fig. 2.7. Each link is full-duplex mode with equal bandwidth. Each connection originates from nodes on the left, and ends on the nodes on the right. Acknowledgment packets do not interfere with the data packets on the forward path.

In Fig. 2.7, 320 flows' source and destination location is shown. Flows are set up between the node pairs with same flow ID. Flows whose ID is greater than 320 are assigned similarly. Up to 800 TCP flows were simulated. Each queue's maximum size is 120 packets. The TCP flows are assumed to be FTP connections with infinite data to send. The packet size is 1 KB. They adapt their sending rate according to TCP-Reno specification[18].

For RED, maximum marking probability max_p is set to 0.1 and minimum threshold min_{th} is 30. Maximum threshold max_{th} is set equal to the maximum queue size 120. This is done so as to obtain results driven by early detection of congestion. If the early detection algorithm fails, packet loss occurs. The filter coefficient for the queue w_q was set to 0.005. For TMCA, target queue size q^* was set to 60. These parameter settings are used in the sequel unless otherwise specified.

We first presented results for a case where the number of flow alternate between 80 and 40. This was done to examine the impact of such changes on performance. During a simulation time of 160 seconds, 40 flows remain up, while other 40 flows are turned on and off every 40 seconds.

The link bandwidth and propagation delay are exhibited on the links in Fig. 2.7. There are two bottleneck links between node 4 and 5 with 45Mb/s and between 5 and 6 with 10Mb/s. All other links are of 100Mb/s. The link propagation delays vary from 1 ms to 5ms, 10ms and 15ms. So each flow's RTT ranges from 22 ms to 80 ms.

Table 2.2: Bottleneck link performance (N4-N5).

type	Throughput(Mbs)	Loss Ratio
RED	44.326	0.001 (95% c.i. [0.00093 , 0.00107])
TMCA	44.826	0.00049 (95% c.i. [0.000438 , 0.000542])

c.i. : confidence interval

The performance comparison is given in Table 2.2. The data is averaged over the whole period of operation [0,160]. The throughput is calculated at the bottleneck link [n4,n5] as the total sum of packets serviced over total simulation time. The loss ratio is also calculated at the bottleneck link [n4,n5] as lost packets over lost packets plus serviced packets. The confidence intervals are calculated with the assumption that each packet loss is independent. Packet size is 1 KByte. The number of packets transmitted in [0,160] is approximately $44\text{Mbits}/10\text{Kbits} * 160 = 160 * 4400 \simeq N$. Given loss ratio p_e in the Table 2.2, 95% confidence interval can be found using the following formula [19],

$$p_e \mp 1.960\sqrt{\frac{p_e(1-p_e)}{N}}$$

TMCA exhibits better performance in both packet loss ratio and throughput.

A representative sample path for the queue's behavior for each case is shown in the time charts Fig. 2.8 and Fig. 2.9. As can be seen, TMCA has a better control over the queue behavior. RED exhibits excessive rate slowdown when the number of flows is 40. In turn the associated queue is 'closer' to the zero and thus the throughput is reduced - the system is under-utilized.

The second result we present here covers a wider range of flow numbers. In RED, the maximum marking probability max_p is supposed to be tuned by operators. Bigger value of max_p means more aggressive marking. As mentioned earlier, finding the best marking probability is not an easy task. We will try three representative cases 0.02, 0.1 and 1.0. The case 1.0 represents the extreme case of aggressive marking and 0.02 represents a conservative marking, while 0.1 is a typical case in between.

Six cases associated with different numbers of flows (40, 80, 160, 320, 640 and 800) we simulated on the same topology considered previously. The propagation delay setting remained the same. Link capacity for link [n4,n5] and [n5,n6] are linearly adjusted according to the number of flows so that they can carry 0.25Mb/s per flow. All other links have capacity 622Mb/s. The results are given in Fig. 2.10 and Fig. 2.11. The simulation time was 1000 sec, contended for bandwidth during which the same number of flows.

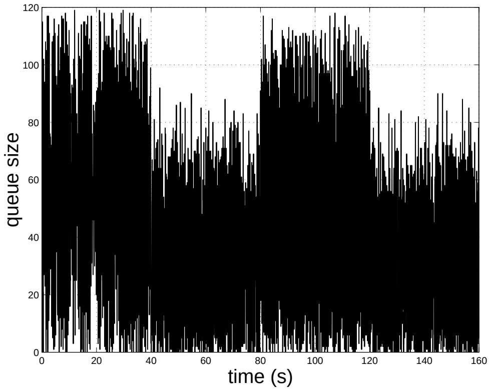


Figure 2.8: RED queue size (max_p=0.1).

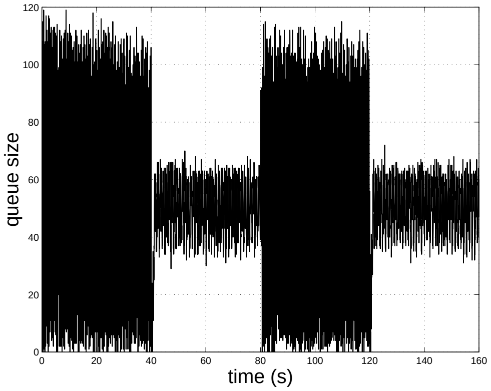


Figure 2.9: TMCA queue size (q^*=60).

Fig. 2.10 shows that conservative marking of RED (max_p=0.02) achieves the best throughput. By contrast Fig. 2.11 shows that aggressive marking gets the smallest loss ratio for RED. Note that with one fixed value max_p in RED, the two performance cannot be achieved at the same time. On the other hand, TMCA achieves high throughput and low loss ratio at the same time. TMCAr, i.e. TMCA with random-delayed rate slowdown, shows about the same performance as TMCA.

The third experiment has the same conditions as the second experiment except for settings for the propagation delays. The delay is set to 5msec for links

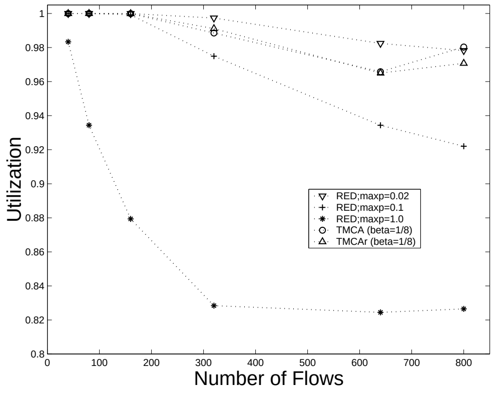


Figure 2.10: Utilization.

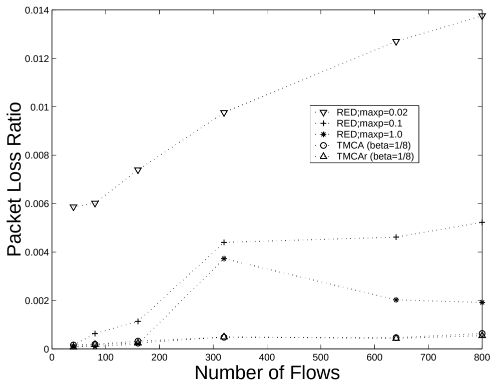


Figure 2.11: Packet loss ratio.

[n4,n5],[n5,n6] and [n5,n7]. For the rest of the links delay is set to 15msec. Note that all flows' round trip propagation delays are now the same (80msec). In this condition, the global synchronization is more significant than the previous case. Each flow's reaction to the congestion notification tends to work in phase.

The results are given in Fig. 2.12 and Fig. 2.13. As can be seen, the performance now deteriorates for because of the intensified global synchronization. Especially in the case of RED, the performance degradation is higher. TMCA (with $\beta = \frac{1}{8}$) still shows better performance than RED. Here we see that we can

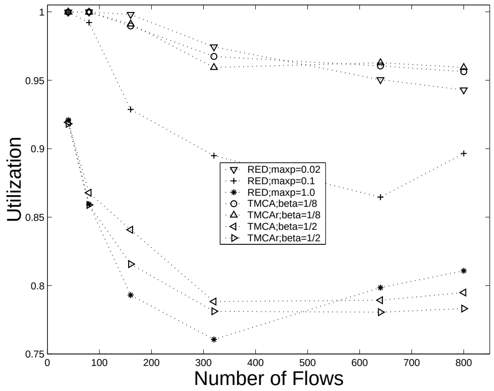


Figure 2.12: Utilization.

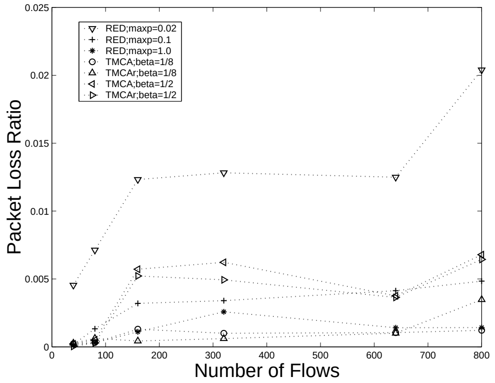


Figure 2.13: Packet loss ratio.

improve the performance further by using TMCAr. We included the case of $\beta = \frac{1}{2}$ for comparison. First we can see the performance is not good and the TMCAr's function is excessive when β is $\frac{1}{2}$. TMCAr(with $\beta = \frac{1}{2}$) exhibits lower packet loss ratio than TMCA(with $\beta = \frac{1}{2}$). Note that the TMCAr's utilization is however also lower than the TMCA. Thus, when the rate variance is big and queue size is not large enough to support it, TMCAr(e.g. $\beta = \frac{1}{2}$) can aggravate under-utilization. This is the case because dispersed big decreases (with $\beta = \frac{1}{2}$) in rate lengthen the under-utilization time period. To remove this problem, we need to increase the buffer size and adjust q^* level accordingly. When β equals $\frac{1}{8}$, TMCAr shows about the same or better result than TMCA and TMCAr's contribution to the performance is less significant. In this case, we can see that threshold-based marking scheme with reduced rate decrease has much bigger impact on the performance than the random-delayed slowdown. The amount of delay in random-delayed slowdown affects the average of total arrival rate since rate stays at a higher rate longer prior to decreasing. This helps explain why the improvement in performance from TMCA to TMCAr is not as big as the improvement of TMCA over RED.

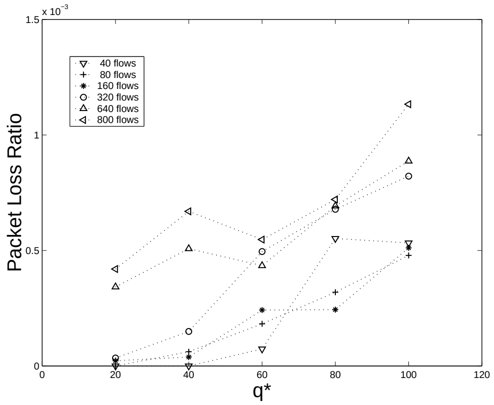


Figure 2.14: Packet loss ratio (TMCAr).

The final experiment we conducted was to investigate the dependence of performance variables, such as packet loss ratio, throughput and queue average, on the queue threshold q^* in TMCAr. The parameter setting here are the same as the second experiment. We used various levels for q^*. The results are exhibited in Fig. 2.14, Fig. 2.15 and Fig. 2.16. In the same experimental conditions, TMCA shows similar performance.

TMCA and TMCAr can maintain relatively a steady low loss ratio when q^*

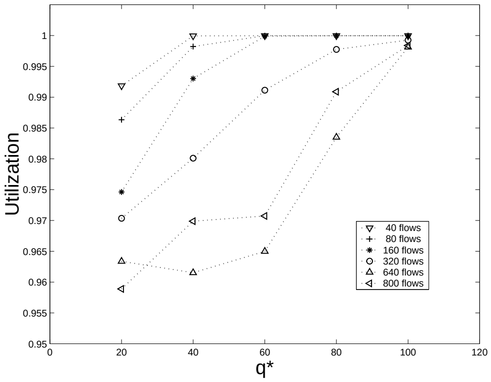


Figure 2.15: Utilization (TMCAr).

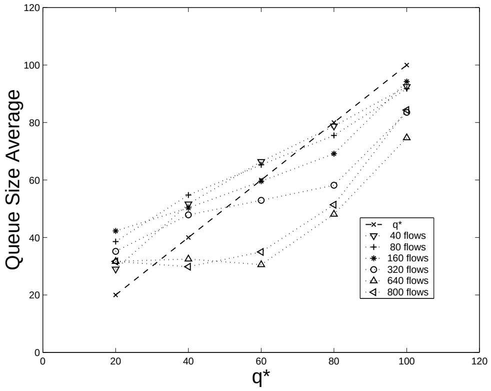


Figure 2.16: Queue size average (TMCAr).

is in the region of 20 to 60. In the region of 60 to 100, loss ratio increases, and the utilization is flat and high when the flow number is 160 or less. For 320 flows or more, utilization increases as q^* increase beyond 60. In other words, to get a higher utilization it is necessary to increase the queue threshold in order to cover larger rate variance. The queue average plot in Fig. 2.16 shows the queue average can be controlled to a lower value by reducing q^*. Lower queue average means reduced average queueing delay. Fig. 2.15 and Fig. 2.16 show that reducing queueing delay by decreasing q^* means compromising the throughput.

2.5 Conclusions

We tested the performance of our proposed threshold-based binary feedback (TMCA) and window-based flow control and with that of RED. In our comparison, we noted that a modification to the multiplicative decrease of TCP was necessary. We explored their performance with wide ranges of parameter settings. We found that TMCA shows superior performance over RED. TMCA is simple and does not need to 'estimate' the marking probability. Moreover, generation of random markings would not be required at routers. By adding random-delays to the rate reduction component (TMCAr) at end hosts, we can improve the performance further. The increased complexity in TMCAr must be traded-off against the improvement in performance which becomes less significant as the rate reduction ratio gets small.

We have provided guidelines to set q^* to half of the maximum queue size. When queue length varies symmetrically and closely around q^*, near-maximum throughput and near-minimum loss ratio can be achieved. Further reduction in queueing delay is possible by compromising the throughput achieved.

ECN is an experimental protocol which requires modifications of the protocol at end hosts for full standardization. The work presented here provides motivations to further explore the rate decrease ratio for TCP at the end hosts.

Chapter 3

Traffic engineering minimizing congestion of both traffic load and signaling loads

3.1 Introduction

Although network capacity has been increasing quickly to keep the pace with increasing demands, the unpredictability of growth, and bursty nature of traffic not to speak of costs (particularly at the network edge) invariably result in networks with congestion. Congestion occurs when the network has insufficient resources to support the offered load. Specifically congestion at a given point (i.e. link or node) in the network can be defined as a function of two parameters; the amount of resources currently used and the maximum resource capacity of the point. We refer to a certain point under heavy congestion in the network, as a *hot spot*.

In response to growing network traffic, one can either increase the capacity of the network or re-distribute the load. The capacity and traffic management are two fundamental problems addressed in traffic engineering (TE)[20].

Our work in this chapter focuses on network routing. We assume shortest path routing (SPR) as our basic framework. SPR is a well-proven, easy to implement and widely-used scheme. There are many well known algorithms, e.g. Dijkstra's, to determine shortest path routes. In current routing implementations, e.g. PNNI[37] (for ATM networks) and OSPF[22][23] protocol (for Internet), SPR is used. In SPR, each link is associated with a value called *link metric*. Link metric is interpreted as the cost to traverse the link. SPR selects a path that has the smallest link metric sum. In practice networks, link metrics are commonly set to a same 'administrative' weight value, in which case the routed path implements a minimum-hop route. According to the system objective one wishes to optimize, there are many ways in which one could set the administrative weight. For example when the transmission delay is the primary consideration, the cost at link l can be set to $1/C_l$, where C_l is the link capacity.

Using the same constant value for all links or $1/C_l$ are easy ways to set the administrative weight associate with each links. However, since these are only based on link-specific information, the information about the traffic loads is not factored in. In multi-service networks, traffic demand distribution tend to be both temporally and spatially diverse. Information about the traffic load may be available through measurements over a certain period of time or through customer contracts[21]. By incorporating the traffic demand information into the problem formulation, we can identify hot spots and reduce congestion. There are many proposed schemes for setting administrative weights that include both the link information and the traffic demand[24][25].

In much of the traffic engineering research/technical literature, bandwidth guarantees are commonly discussed as the primary concern and problems are formulated with a focus on bandwidth allocation. Other performance measures such as packet loss or delay are typically believed to be effectively handled by using the effective bandwidth framework[65]. Since delays increases substantially as the offered load reaches the capacity limit[35], the primary interest is to evenly distribute bandwidth loads throughout the network. Of course in order to accommodate more traffic, it is also important to minimize the total bandwidth consumption. In this chapter we shall refer to the bandwidth capacity as the *bandwidth resource*.

Increasingly networks are merging and supporting multiple services. In this environment, in order to achieve the desired QoS services are becoming increasingly connection-oriented or supported over overlay networks. The bandwidth requirement of various application's connections tends to be very diverse. At the same time, regardless of the amount of bandwidth that needs to be dedicated a certain amount of connection processing and maintenance is commonly required. For billing and Service Level Agreement(SLA) purposes, CPU capacity and memory space are required to process and manage necessary data on a per-connection basis. Also connection setup requires processing of the necessary signaling required[26]. Moreover, depending on the algorithm complexity, routing will require a certain amount of computation and processing. Due to limited processing resources to handle such traffic, there is an upper bound on the amount of signaling maintenance traffic the network can sustain.

In connection-oriented multi-service networks based on MPLS or PNNI, a high level of reliability is required. Thus the ability to restore routes promptly upon failure is a very important issue. The survivability of a network gauges the percentage of traffic upon a single failure. To provide a guarantee on survivability of a network, enough bandwidth needs to be reserved for backup paths. On the other hand, to provide fast enough restoration speed, it is crucial to maintain a small number of connections on each link throughout the network. Indeed since restoration actions are performed on per-connection basis, the target restoration speed sets an upper bound on the number of connections a link can carry. In this case the upper bound is only on the connections which require restoration. In general the lower the number of connections per link the higher the speed to restore them upon failure.

In some cases, the technology imposes a hard limit on the number of connections per link. In ATM networks for instance, up to 4096*65536 connections can be set up on a given NNI(Network to Network Interface) link. However in practice real ATM switches may not be able to support that much activity per link. When a switch is designed, for a given bandwidth capacity, optimal CPU and memory capacity are determined by assessing expected connection processing load and other requirements. If the load is beyond the designed processing range, the operating system performance of the ATM switch degrades very rapidly. Designers usually set a maximum limit on the number of connections to avoid such degradation. Connection setup requests beyond that level are either simply rejected or re-directed to other possible links through crankback. Because of these reasons, a limit of 4000 connections per OC3 port are not uncommon in the real world. As increases in bandwidth capacity result in higher costs, so will increase the maximum number of connections that can be supported by a given switch design. We refer in this chapter to the connection processing capacity limits as *connection signaling resources*.

We have argued that the connection signalling resource is an important constraint to be carefully considered in multi-service networks especially when the system supports large numbers of 'low' bandwidth connections. If we incorporate the connection signalling resource constraints into the routing framework, we can potentially reduce blocking and/or unnecessary crankback of connections caused by connection signalling resource shortages. Additionally we can enable faster restorations.

By explicitly accounting for connection signalling resources, we can model multi-service network more accurately. The requirements on connection signalling resources are readily available in the contract information as connections are set up or can be obtained through measurement. The unit of the connection signalling load can be either the count of ongoing connections or the count of connections setup per limit time. The former captures 'static'(i.e. maintenance overheads) resource requirements while the latter captures dynamic resource requirements. In our work, we place resource constraints on the number of connections a link/node can support. The problem for the case of dynamic resource requirements is similar. Finally when we consider restoration, the number of connections translates to a potential dynamic load upon failure.

The basic idea of minimum congestion routing is to guide the traffic around hot spots and therefore reduce the cost that originates from congestion. One can set up an optimization problem to find flow solution that minimizes overall congestion cost. The congestion cost is not simply proportional to the amount of used resource. Once a part of a network is congested, increased crankback can cause detrimental impact on the other side of the network. And as the amount of used resource approaches its resource capacity, we observe quick performance drop in real systems. A nonlinear convex cost can be a reasonable way to model the load-cost relation. More detailed discussion on the network congestion will follow in the later section.

Once the optimum flow solution is found, next challenge is to find a collection of fixed link metrics such that SP routing would achieve a routing of traffic which is close to optimum. We will propose an effective algorithm to compute link metrics.

When there are more than one shortest paths between any source destination pair, tie-breaking rule and load distribution ratio need to be specified. We assume that we distribute load on the all the shortest paths of each source-destination pair and enforce even splitting of loads at branch nodes on the equi-distance paths.

In related work, Wang et al.[24] set up an optimization problem for a single resource type and propose an algorithm to set the link metrics. A primal-dual problem is used to obtain the necessary weights. They work with traffic which is of homogeneous commodity. By homogeneous, we mean the case when each connection's required bandwidth resources are all equal. We take a similar approach with multiple resource notion, which formulates network problems with nonhomogeneous commodity traffic.

Fortz et al.[25] propose a closed loop architecture where feedback on routing performance is used to adjust the link metric set. Their approach uses an off-line algorithm, in which each iteration stage uses Djikstra's algorithm on the current link metric set to get routing performance. Based on the result, the link metric set is updated and so forth. One of the limitations of their approach is that it can get trapped in local optimum. For this reason they propose a perturbation mechanism, however it can hardly guarantee the convergence to an optimum. Their approach requires an excessive number of iterations.

Kodialam et al.[30] study how to evade topologically critical links (which is similar to hot spots in our work) so that a newly routed tunnel must follow a route that does not interfere too much with a route that may be critical to satisfy a future demand. Their work is in the dynamic on-line routing framework and each path is set up through source routing.

The contribution of this chapter is as follows. First, we set up a network flow optimization problem with multi-resource constraint incorporating the notion of connection signalling resource. Second, we propose a new algorithm to calculate the optimal link metric that keeps the flow distribution realized by SP routing close to the optimal solution flows. The algorithm can be implemented off-line and the generated link metrics has immediate applicability. The assumed even splitting on equi-distant paths is commonly available in networks or simple to add. Without adding any additional parameters to the current routing framework, i.e. by assigning only a new link metric set, congestion cost is substantially reduced. According to the availability of the traffic information, proposed algorithm can be used in a quasi-static manner, e.g. modified depending on the time of day or weeks.

The remainder of this chapter is organized as follows. The following section provides a detailed discussion on congestion and the multi-resource constraints. In Section 3.3, we formulate an optimization problem and explain the steps we take to determine the final link metrics. We then carry out numerical experiments with a sample topology. In Section 3.4 the results and discussions are provided. The Section 3.5 concludes the chapter.

3.2 Minimum congestion routing

3.2.1 Congestion

Congestion occurs when a network's spare resources become scarce. The congestion at a certain link will be a function of the resource that are currently used and resource capacity of the link. We will use congestion and congestion cost interchangeably in this chapter.

Utilization could be used to represent congestion cost. But it increases linearly as the used resource approaches capacity limit. We observe the overall performance deteriorates drastically near capacity limit because of the aggravated crankback or delayed processing. It is reasonable to assign more cost as a penalty as the utilization on a link approaches 1.0. Any monotone increasing convex function may be a good way to model the quickly-increasing cost. Quadratic curve can be

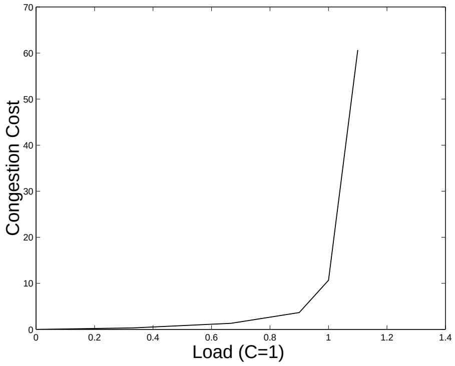


Figure 3.1: $\Phi(l, c)$: congestion cost function [Fortz et al.].

one choice. To formulate network flow problem in linear programming for solving convenience, we prefer the function to be approximated using piece-wise linear convex function. Fortz et al. [25] proposed such a function and we found that it is indeed appropriate for our needs. It is given in (3.1) where l denotes load and c is the capacity of a link. The piece-wise linear function is as shown in Figure 3.1. The slope of the cost increases quickly from 1 to as high as 5000 above the full utilization. Note that this is a function of both the load and capacity, not simply the utilization, i.e. the ratio. This serves to differentiate the link when their capacities are different. The bigger link with the same utilization gets assigned bigger cost.

$$\Phi(l,c) = \begin{cases} l & \text{if} \quad 0 < l < \frac{1}{3}\,c \\ 3l - \frac{2}{3}c & \text{if} \quad \frac{1}{3}\,c \le l < \frac{2}{3}\,c \\ 10l - \frac{16}{3}c & \text{if} \quad \frac{2}{3}\,c \le l < \frac{9}{10}\,c \\ 70l - \frac{178}{3}c & \text{if} \quad \frac{9}{10}\,c \le l < c \\ 500l - \frac{1468}{3}c & \text{if} \quad c \le l < \frac{11}{10}\,c \\ 5000l - \frac{16318}{3}c & \text{if} \quad l \ge \frac{11}{10}\,c \\ 0 & \text{if} \quad l = 0 \ \text{ or } \ c = 0 \end{cases} \tag{3.1}$$

In our problem formulation, we are interested in minimizing the *overall congestion*. Using the cost function model at each link as above, there are many alternatives to define the overall congestion in a network. We let Φ denote the congestion cost function, l_k denote the link load and c_k the link capacity. Here we discuss three possible definitions. They are maximum of congestion at any link

$$\max_{k \in \{\, links\}} \left(\ \Phi(\ l_k\ ,\ c_k\ )\ \right), \tag{3.2}$$

the summation of the resource currently used

$$\sum_{k \in \{\, links\}} (\ l_k\ ), \tag{3.3}$$

and the summation of congestion at the links in the network

$$\sum_{k \in \{\, links\}} (\ \Phi(\ l_k\ ,\ c_k\ )\ ). \tag{3.4}$$

Let us consider a case where the maximum congestion value (Eq. (3.2)) is used as the objective function. In this case the problem is of the Min-Max format. We can directly control the maximum utilization level in the network. The limitation of this method is that it only considers the maximum congestion, i.e. the spread of the load on other links is not strictly controlled. Thus for example the number of hops a route takes is not limited. As long as the maximum congestion is minimum, long paths are not detected as causing more congestion. Note however that long paths increase the overall resource consumption and therefore may hinder other paths from using network resources.

If instead the total sum of all link's resource consumption (Eq. (3.3)) is taken as the definition of the overall congestion, this might serve to reduce the total utilization. Unfortunately such a cost function does not differentiate the case of uneven utilization from the case of a well-balanced utilization. For instance, consider a case when one distribute traffic which is of amount 1.0 over two paths. If we follow the definition in Eq. (3.3)), the distribution of amount 0.01 and 0.99 on the two paths becomes of the same cost as the case where traffic amount 1.0 is distributed to be 0.5 and 0.5 on the paths even though the latter case is more preferable.

If we choose an overall network congestion cost as in Eq. (3.4), the problem with the previous cases can be cleared. Excessively long paths can be filtered out and even load distributions can be achieved.

If this summation of convex cost is used as the objective function in the network flow problem, we get advantages. It is a good combination of resource usage minimization and high cost assignment to the link in heavy utilization. When the offered load is light, solution flows are directed in a similar way as in minimum hop routing.

3.2.2 Multiple resources

To use congestion function in (3.1) for multiple resources, one need to be able to scale their relative importance. scaling issue should be resolved. From (3.1) it follows that the following scaling relationship holds with $\alpha > 0$,

$$\alpha \, \Phi(\, l \; , \; c \,) \; = \; \Phi(\, \alpha \, l \; , \; \alpha \, c \,). \tag{3.5}$$

It follows that the congestion cost of various resources can be re-scaled to handle resources with different capacities. A possible idea is to pick the maximum resource level for each type and scale the cost so that they are equal. Let $C_{(i,j),\,bw}$ and $C_{(i,j),\,cn}$ to be bandwidth capacity and connection capacity of link (i, j) respectively. And let L be the set of links. By introducing scaling coefficients α_{bw} and α_{cn}, we scale the costs for different resources as shown in (3.6) with a constant C^*.

$$\alpha_{bw} \max_{(i,j)\in L} (C_{(i,j),\,bw}) = \alpha_{cn} \max_{(i,j)\in L} (C_{(i,j),\,cn}) = C^* \tag{3.6}$$

For example, suppose a network contains OC3 links with a maximum of 4000 connection capacity and OC12 links with a maximum of 16000 connection capacity. We can scale them by adjusting α_{bw} and α_{cn} and making the cost of bandwidth 622.08

Mb/s and the cost of 16000 connections to be the same. We will use this idea in our problem formulation.

3.3 Modelling and algorithm

The notations which we use for modelling are summarized in Table 3.1. Links are assumed to be directed, which means link (i, j) and link (j, i) are treated as separate entities. Each link has its own link metric and capacity.

Table 3.1: The notation table.

Notation	Meaning	Unit
V	Set of Nodes	unitless
E	Set of Links $\subset V \times V$	unitless
K	Set of s-d pairs $\subset V \times V$	unitless
$\Phi(l, c)$	Congestion Cost Function	unitless
$X_{(i,j)}^{(m,n)}$	Flow Variable, $(i, j) \in E, (m, n) \in K$	cn
$b^{(m,\,n)}$	Offered Load on $(m, n) \in K$	cn
$\beta^{(m,\,n)}$	Bw Requirement of $(m, n) \in K$	Mbps/cn
$C_{(i,j),\,bw}$	Link Capacity (Bw)	Mbps
$C_{(i,j),\,cn}$	Link Capacity (Cn)	cn
α_{bw}	Scaling coeff. (Bw)	1/Mbps
α_{cn}	Scaling coeff. (Cn)	1/cn
$L_{(i,j),\,bw}$	Resource Consumption (Bw)	Mbps
$L_{(i,j),\,cn}$	Resource Consumption (Cn)	cn
$\Phi_{(i,j),\,bw}$	Congestion Cost Variable (Bw)	unitless
$\Phi_{(i,j),\,cn}$	Congestion Cost Variable (Cn)	unitless
$X_{(i,j)}^{*(m,n)}$	Optimal Sol. for Min. Congestion	cn
$\hat{C}_{(i,j),\,bw}$	Tight Bound Capacity (Bw)	Mbps
$\hat{C}_{(i,j),\,cn}$	Tight Bound Capacity (Cn)	cn
$U_{(k)}^{(m,n)}$	Dual Variable at node k	unitless
$W_{(i,j)}$	Dual Variable on link (i,j) (Bw)	unitless
$Z_{(i,j)}$	Dual Variable on link (i,j) (Cn)	unitless
$A_{(i,j)}^{(m,n)}$	Admin. Weight on link (i,j) for (m,n)	unitless
$\tilde{A}_{(i,j)}$	Heuristic Admin. Weight on (i,j)	unitless

(cn: Connections)

In this chapter, we deal with the static routing, and we assume prior knowledge on the network traffic matrix (i.e. offered load) is available. This assumption is one of the most intensively debated issues in the related literature, see e.g. [21]. We assume the offered load information is available through monitoring of traffic statistics. For example if routing is done on a time-of-day or the day-of-week basis, one can forecast the periodical traffic load. Based on such information, routing performance can be improved.

The eventual goal in this section is to find a link metric set for SP routing that can give 'optimal' performance. Let us present a brief outline of the steps we will follow to reach the goal in this section.

We first formulate a problem with the multi-resource notion and overall congestion function discussed in the previous section. From the solution we get by solving the problem, we try to get the link metric set. Our approach to this is to use the complementary slackness condition (CSC) of linear programming (LP) problem[36]. To utilize CSC, we formulate a derived network problem using the flow solution we already got. From the primal-dual relation of this simple problem, metric set can be found. More detailed explanation will follow in the sequel. Since the current SP-based routing framework uses only one set of link metrics (when multi-class is supported, at most the same number of metric sets as the number of classes would be required), whence we need to reduce this to a single collection of link metrics. We propose a heuristic approach to find a single set. The proposed algorithm is summarized at the end of this section.

3.3.1 Minimum congestion optimization problem

With multiple resources in mind, we formulate a new optimization problem in which both bandwidth and connection resources are considered. The goal is to reduce the overall congestion discussed in the previous section. More specifically, the objective is to minimize the total sum of congestion cost on the links.

Minimize

$$\sum_{(i,j)\in L} \Big(\alpha_{bw}\Phi(\ L_{(i,j),\,bw}\ ,\ \ C_{(i,j),\,bw}\ ) + \alpha_{cn}\Phi(\ L_{(i,j),\,cn}\ ,\ \ C_{(i,j),\,cn}\ ) \Big) \tag{3.7}$$

subject to

$$L_{(i,j),\,bw} = \sum_{(m,\,n)\in\ K} \beta^{(m,\,n)} X_{(i,j)}^{(m,\,n)} \qquad (i,j)\in E \tag{3.8}$$

$$L_{(i,j),\,cn} = \sum_{(m,\,n)\in\ K} X_{(i,j)}^{(m,\,n)} \qquad (i,j)\in E \tag{3.9}$$

$$\sum_{j:(m,j)\in\ L} X_{(m,j)}^{(m,n)} - \sum_{k:(k,m)\in\ L} X_{(k,m)}^{(m,n)} = \ b^{(m,n)} \quad (m,n)\in K \tag{3.10}$$

$$\sum_{j:(n,j)\in\ L} X_{(n,j)}^{(m,n)} - \sum_{k:(k,n)\in\ L} X_{(k,n)}^{(m,n)} = -\ b^{(m,n)} \quad (m,n)\in K \tag{3.11}$$

$$\sum_{j:(i,j)\in\ L} X_{(i,j)}^{(m,n)} - \sum_{k:(k,i)\in\ L} X_{(k,i)}^{(m,n)} = \ 0 \qquad (m,n)\in K\,,\ i\neq m\,,\ i\neq n\,,\ i\in V \tag{3.12}$$

$$X_{(i,j)}^{(m,n)} \ \geq\ 0 \qquad (i,j)\in L\,,\ \ (m,n)\in K \tag{3.13}$$

Let L be the set of links. The bandwidth and connection signalling resource capacity of a link (i, j) are denoted by $C_{(i,j),\,bw}$ and $C_{(i,j),\,cn}$. Expression (3.7) shows the objective function. We scale the load on the two resources so that we can add those values. Equation (3.8) and (3.9) capture the resource consumption used in (3.7). The set K denotes the set of source-destination(s-d) pairs and the coefficient $\beta^{(m,\,n)}$ denotes the bandwidth requirement of an s-d pair (m, n). We assume that the bandwidth requirement of the connections from the same s-d pair are equal and otherwise the bandwidth requirement of a connection is different for different s-d pairs. Multiple types of connections with different bandwidth requirement on each s-d pair can be covered by attaching pseudo nodes to the original s-d nodes and re-assign respective offered loads. The coefficients α_{bw} and α_{cn} in (3.7) are defined in Eq. (3.6). Flow conservation constraints are given in (3.10), (3.11) and (3.12). Eq. (3.13) ensures that flows remain non-negative. We allow splitting(i.e. bifurcation) of each flow. In the case when the offered load consists of a large number of connections, we have enough granularity to split the flow according to the required splitting ratio. Therefore, we can exclude integer condition for the flow variable $X_{(i,j)}^{(m,\,n)}$ in the current problem setup.

The function $\Phi(\cdot)$ is a nonlinear function as it is. To use in LP programming packages, the formulated problem above needs to be modified so that it contains only linear functions. The converted representation in full linear programming problem format is provided in Appendix A.1. We call this problem as MCOP and get flow solution from this piecewise linearized LP problem.

3.3.2 Derived problem with tight constraint

The flows found as a solution to the problem MCOP in the previous subsection form paths from each source to each destination. From this section we devise a technique to make most of these paths correspond shortest paths with respect to static admin weight. In other words, we need to find a way to set link metrics so that the solution paths can be shortest paths.

For the homogeneous commodity case where all the connection has the requested bandwidth of same size, the work by Wang et al.[24] proposes a way to get such link metric set. Since our problem is of non-homogeneous commodity case where corresponding requested bandwidth sizes of connections from various s-d pairs are different, we cannot use the technique directly. But, by taking a similar approach, we can devise a link metric setting algorithm.

Let $X^{*(m,n)}_{(i,j)}$ be the optimal connection signalling resource solution flow on a link (i,j) for s-d pair (m,n) of the problem MCOP. We get the total connection signalling load on each link by summing up the solution flows which pass through the link. In similar way we get the total bandwidth resource load on each link by multiplying the requested bandwidth amount $\beta(m,n)$ for s-d pair (m,n) to the solution flows and summing them all on the link (in connection unit). We denote these sums as $\hat{C}_{(i,j),\,cn}$ and $\hat{C}_{(i,j),\,bw}$. We use these total loads on each link as tight constraints in another problem we formulate now. The key motivation is to utilize the the complementary slackness condition (CSC) of linear programming (LP)[58].

In objective function we now have the total resource consumption (i.e. sum of bandwidth and connection signalling resources used on the all links). With the same offered load at each s-d pairs as in the MCOP and the new tight constraints, we formulate a new derived optimization problem(DPTC) as follows.

$$\text{Minimize} \qquad \sum_{(m,n)\in K,(i,j)\in L} (1\ +\ \beta^{(m,n)})\, X_{(i,j)}^{(m,n)} \tag{3.14}$$

subject to

$$\sum_{j:(m,j)\in\ L} X_{(m,j)}^{(m,n)} - \sum_{k:(k,m)\in\ L} X_{(k,m)}^{(m,n)} =\ b^{(m,n)} \qquad (m,n)\in K \tag{3.15}$$

$$\sum_{j:(n,j)\in\ L} X_{(n,j)}^{(m,n)} - \sum_{k:(k,n)\in\ L} X_{(k,n)}^{(m,n)} = -\,b^{(m,n)} \qquad (m,n)\in K \tag{3.16}$$

$$\sum_{j:(i,j)\in\ L} X_{(i,j)}^{(m,n)} - \sum_{k:(k,i)\in\ L} X_{(k,i)}^{(m,n)} =\ 0$$

$$(m,n)\in K\,,\ i\neq m\,,\ i\neq n\,,\ i\in V \tag{3.17}$$

$$\hat{C}_{(i,j),\,bw} = \sum_{(m,\,n)\in\ K} \beta^{(m,\,n)} X^{*(m,\,n)}_{(i,j)} \qquad (i,j)\in L \tag{3.18}$$

$$\hat{C}_{(i,j),\,cn} = \sum_{(m,\,n)\in\ K} X^{*(m,\,n)}_{(i,j)} \qquad (i,j)\in L \tag{3.19}$$

$$\sum_{(m,\,n)\in\ K} \beta^{(m,\,n)} X^{(m,\,n)}_{(i,j)}\ \le\ \hat{C}_{(i,j),\,bw} \qquad (i,j)\in L \tag{3.20}$$

$$\sum_{(m,\,n)\in\ K} X^{(m,\,n)}_{(i,j)}\ \le\ \hat{C}_{(i,j),\,cn} \qquad (i,j)\in L \tag{3.21}$$

$$X_{(i,j)}^{(m,n)}\ \ge\ 0 \qquad (i,j)\in L\,,\ (m,n)\in K \tag{3.22}$$

Expression (3.14) shows the objective function. Flow conservation constraints are given by (3.15), (3.16) and (3.17). Eq. (3.18) through (3.21) defines tight constraints. Eq. (3.22) ensures non-negativity of the optimal flows.

Note that we do not have α_{bw} and α_{cn} in the objective. Since the congestion cost function is strictly increasing when $X_{(i,j)}^{(m,n)} \geq 0$ and offered load at s-d pairs is same with the tight constraints, the optimal solution in the MCOP can also be optimal solution of this optimization problem.

Now we formulate a dual problem of the DPTC. We do not solve the DPTC. Instead we solve the dual problem and then by using the CSC between the primal-dual pair we find the optimal link weight set. Especially the fact that (3.18) and (3.21) gets binding constraints for the optimal solution in DPTC will be utilized.

3.3.3 Dual problem formulation for the derived problem with tight constraint

To formulate the dual problem d-DPTC of the DPTC we introduce three groups of dual variables. They correspond various constraints in the DPTC. Specifically we introduce dual variables $U_k^{(m,n)}$ for (3.15), (3.16) and (3.17); $W_{(i,j)}$ for (3.20); and $Z_{(i,j)}$ for (3.21).

The dual problem is shown in (3.23)-(3.26) below. To ensure the problem is bounded, we set upper limits on the variables $W_{(i,j)}$ and $Z_{(i,j)}$.

Maximize

$$\begin{aligned}\sum_{(m,n)\in K} b^{(m,n)} U_m^{(m,n)} - \sum_{(i,j)\in E} \hat{C}_{(i,j),cn} Z_{(i,j)} \\ - \sum_{(i,j)\in E} \hat{C}_{(i,j),bw} W_{(i,j)}\end{aligned} \tag{3.23}$$

subject to

$$1 + Z_{(i,j)} + \beta^{(m,n)}(1 + W_{(i,j)}) - U_i^{(m,n)} + U_j^{(m,n)} \geq 0$$

$$(m,n) \in K \; , \; (i,j) \in E \tag{3.24}$$

$$0 \;\leq\; W_{(i,j)} \qquad\qquad (i,j) \;\in\; E \tag{3.25}$$

$$0 \;\leq\; Z_{(i,j)} \qquad\qquad (i,j) \;\in\; E \tag{3.26}$$

Here (3.23) represents the objective function of d-DPTC, and (3.24) is a constraint on the dual variables. Nonnegativity constraints are given in (3.25) and (3.26).

By using CSC, we can get a link metric that makes the solution paths in primal problem to be shortest paths. These are given as follows.

$$A^{*(m,n)}_{(i,j)} \;=\; 1 \;+\; Z_{(i,j)} \;+\; \beta^{(m,n)} \, (1 \;+\; W_{(i,j)}) \tag{3.27}$$

Note that (3.27) contains $\beta^{(m,n)}$ term which is s-d pair dependent. Therefore if we assign the metric in (3.27) for any s-d pair to the links and find the shortest paths, they coincide with the optimal solution paths of the s-d pair in the primal problem (by the partial duality [58]). Even though these $A^{*(m,n)}_{(i,j)}$ realize shortest paths which are optimal, it requires us to assign multiple of link metrics per s-d pair. The objective for a real system is to find a single shared metric. The reason that the metric we found in (3.27) contains s-d pair dependency is because we are dealing with non-homogeneous commodity case. We need to devise a heuristic method to get a single metric set to use in real system. In the next subsection we propose an approach to this end.

3.3.4 Heuristic dual problem for the derived problem with tight constraint

Recall the the weights $A^{*(m,n)}_{(i,j)}$ found in the previous subsection, if used as link metric, would result in 'optimal' routing, assuming SP routing for each s-d pair (m,n). Recall that the link metric does not contain any information on how much of the offered load should actually be distributed on each equi-distant shortest paths. Our current assumption is that demands would be split evenly among them.

It is important whether the variable $X^{(m,n)}_{(i,j)}$ in optimal solution of DPTC is zero or not. If it is not zero, the link is on the optimal path for s-d pair (m,n). And the expression for the link metric $A^{*(m,n)}_{(i,j)}$ with the dual variable solution $W_{(i,j)}$ and $Z_{(i,j)}$ in (3.27) enables the solution paths to be shortest paths. $W_{(i,j)}$ and $Z_{(i,j)}$ enable multiple optimal paths to be equi-distant in terms of the sum of link metrics $A^{*(m,n)}_{(i,j)}$ along the paths.

Now suppose we have another new problem with only one type of constraint. Let the new problem's solution flows are non-zero on a link if and only if the DPTC's solution flow is non-zero on the link. Then the new problem's optimal path set can coincide with the DPTC's solution paths. The ratio of the flow distribution is not important in setting the link weight. On the other hand, which link's flow is non-negative is important.

As a new problem for a heuristic approach, we propose to formulate one as in (3.28)-(3.30). The formulation only contains a tight constraint on connection resources. Since these constraints are tight, the corresponding non-zero solution flows in the DPTC can also be non-zero. Because the bandwidth constraint is missing, the solution may violate the constraint but still whether the flow $X^{(m,n)}_{(i,j)}$

is zero or not will be preserved in both problems. We denote this new problem as Hd-DPTC.

$$\text{Maximize} \quad \sum_{(m,n)\in K} (b^{(m,n)} U_m^{(m,n)}) - \sum_{(i,j)\in E} (\hat{C}_{(i,j),cn} V_{(i,j)}) \tag{3.28}$$

subject to

$$1 + Z_{(i,j)} - U_i^{(m,n)} + U_j^{(m,n)} \geq 0$$

$$(m,n) \in K \ , \ (i,j) \in E \tag{3.29}$$

$$0 \leq Z_{(i,j)} \qquad (i,j) \in E \tag{3.30}$$

Once again (3.28) is the objective function of Hd-DPTC. Tight constraint $\hat{C}_{(i,j),cn}$ is given as in (3.19). Eq. (3.29) is the constraint for the dual variables and (3.30) ensures the dual variable $Z_{(i,j)}$ is nonnegative. By using CSC, (3.22) and (3.29), we can get a formula for a heuristic link metric $\tilde{A}_{(i,j)}$ as follows.

$$\tilde{A}_{(i,j)} = 1 + Z_{(i,j)} \tag{3.31}$$

Here Eq. (3.31) is independent of s-d pair. It generates a single set of link metrics. Note if we only use the tight constraint on bandwidth instead of connection resource, the metric formula obtained would still depend on the offered load (s-d pair). This is because the flow variable in the original problem is in connection unit. When we use a bandwidth-related term, it is accompanied with $\beta^{(m,n)}$. So it has s-d pair dependency. If we take the bandwidth resource as a variable $X_{(i,j)}^{(m,n)}$

Table 3.2: Link metric setting algorithm.

1. Set up and solve the minimum congestion optimization problem (MCOP).

2. Find the tight constraints for connection resource in the solution of MCOP.

3. Set up and solve the heuristic dual problem (Hd-DPTC).

4. Among the tight constraints found in step 2, find any link whose tight constraint is zero. Assign the maximum link metric (but not ∞) to the link

5. Using the solution $Z_{(i,j)}$ of Hd-DPTC, assign the value $(1 + Z_{(i,j)})$ to the links which have yet been given a metric.

and represent connection resource as $X_{(i,j)}^{(m,n)}/\beta^{(m,n)}$, we can use tight constraint for bandwidth resource as the only constraint and can get link metric formula as in (3.31). In this case the formula will contain $W_{(i,j)}$ rather than $Z_{(i,j)}$.

Let us summarize our proposed algorithm to generate a link metric set. See Table 3.2. It consists of five steps. In the original problem if any solution flow sum on a link is zero, the tight constraint is zero. In this case, as shown in the algorithm, we assign maximum number as a link metric so that it can be classified as up-link. It will not make a difference to set it as a down-link (which means we prune it). But at least they should work as a backup path if a working link fails. Therefore it is more reasonable to set a maximum possible metric to such link.

3.4 Numerical experiments

3.4.1 Topology and test scenario

We experimented with an ATM network with 15 nodes and 28 links as shown in Fig.3.2. The topology is taken from [30]. Our objective is to illustrate the effectiveness of multi-resource notion and the proposed algorithm. The applicability of the proposed framework is not limited to ATM networks.

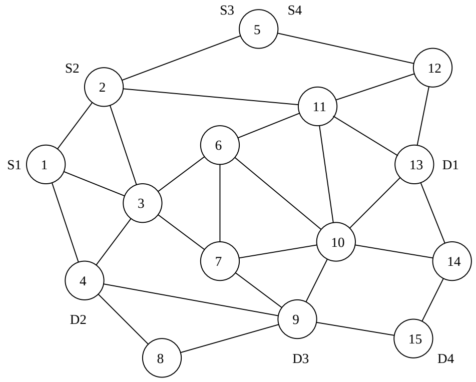


Figure 3.2: The topology with 15 nodes [Kodialam et al.].

We assume there are two types of applications. One type requires 64kb/s bandwidth per connection and the other requires 20kb/s bandwidth per connection. The former represents relatively-bandwidth-intensive application such as VTOA and the latter represents relatively-connection-intensive case such as ADSL best-effort application. We assume all the links have an OC3 bandwidth capacity (155.52 Mb/s) and connection capacity of 4000.

The offered load type and intensity are as given in Table 3.3. The Load between 1 and 13 and the load between 5 and 9 are for connection-intensive applications. Others are for bandwidth-intensive applications. We deal with nine different

Table 3.3: The offered load table.

s-d Pair	$\beta^{(m,n)}$	$b^{(m,n)}$
1-13	0.020	$500x$
13-1	0.020	$500x$
2-4	0.064	$500x$
4-2	0.064	$500x$
5-9	0.020	$500x$
9-5	0.020	$500x$
5-15	0.064	$500x$
15-5	0.064	$500x$

$(x \in [1,9], x : integer)$

load intensity levels (from 500 to 4500 connections). In each level, the connection load intensity is assumed to be the same in the two different applications.

In the topology, node 5 has two links and both of (5,9) and (5,15) s-d pairs originate from the node 5. When the load becomes 4500 connections for each s-d pairs, we see it is beyond the connection resource capacity 4000 of the link (5,2) and (5,12). We assume there is no blocking. We regard the case where the load level is 4500 as overbooking case.

We try three cases of link metric setting algorithm and compare the load distribution results. They are the constant metric case; the bandwidth resource consideration only case; and the multi-resource consideration case.

After we get the calculated link metric sets, we run Dijkstra's algorithm. And we enforce an even split on the equi-distant shortest path routes.

3.4.2 Results and discussion

Among many available linear programming packages, we used AMPL package with CPLEX solver[38]. Results are shown in Figures 3.3 through 3.7. In each figure, the x-axis represents the total load on the network in connection unit.

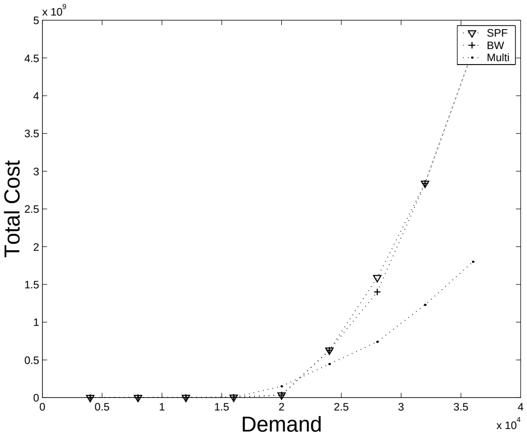


Figure 3.3: Total congestion cost.

To evaluate the performance of the proposed algorithm, we plot the congestion cost value as defined in (3.7). The congestion cost value is the objective function of MCOP. The plot is shown in Fig.3.3.

In the figure, the plot designated by *Multi* shows the proposed algorithm's total cost. The plot with *SPF* legend denotes the case of SP routing with the proposed link metrics. Since all the capacity of the links in the network are the same, default administrative weight such as the reciprocal of the capacity or simply constant weights per link, will not result in any discrimination.

We see our algorithm reduces total cost substantially as the load gets heavier. Since the BW case does not consider the connection-resource, the result does not show significant improvement from SPF case.

Fig.3.4 shows the maximum bandwidth resource utilization across all links. It is easy to see that the proposed algorithm's resource utilization value is not excessive versus the other algorithm. Recall that the curves only represent the

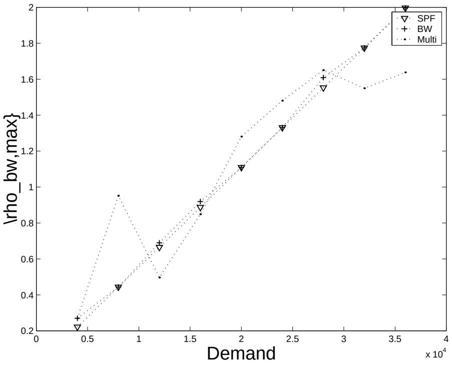


Figure 3.4: Max. utilization (Bw).

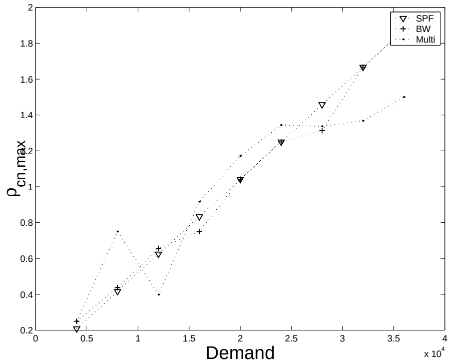


Figure 3.5: Max. utilization (Cn).

maximum bandwidth load value at the worst case. They do not necessarily convey the overall loading in the network. Similarly the maximum connection resource utilization is given in Fig.3.5.

Fig.3.6 and Fig.3.7 shows the resource consumption plots. Resource consumption is defined as the summation of bandwidth or connection resource usage on all the links in the network. The simple shortest path algorithm's optimization objective can be interpreted as the minimization of the bandwidth resource consumption. In that case, the objective is closely related to the total number of hops that the paths take. The plot with the legend *Multi* shows bigger resource consumption as might be expected since the algorithm would take detours path around bandwidth- or connection-resource hot spots. Still the detouring amount is minimized through optimization to keep the total congestion down. Note that excessive detouring can cause congestion on other links.

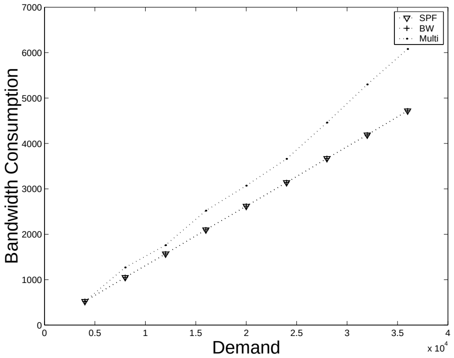


Figure 3.6: Resource consumption (Bw).

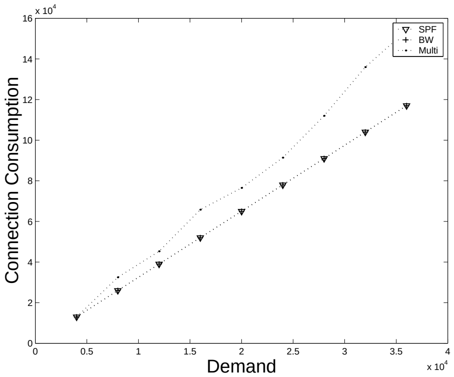


Figure 3.7: Resource consumption (Cn).

Our algorithm is unique in that it addresses the connection resource notion, i.e. accounts for signaling loads throughout the network. It provides link metric set that can deal with connection resource constraint in a quite efficient way. If the parameters in routers and switches are set without consideration on the connection resource, the shortage of connection resource on realized shortest path may cause unnecessary crankback possibly aggravating congestion.

Another advantage of the proposed algorithm is that it does not involve iterative optimization steps. Our algorithm requires only a two step process.

The only assumptions we have made is the even distribution of loads on equi-distant paths. Since it generates one set of link metric, our algorithm is directly applicable to any static routing network with SP framework. There is no need to modify the routing framework or protocol.

Our connection resource notion can be extended to adaptive routing frameworks as well. As addressed above, since connection resource awareness is an important notion for efficient traffic engineering, it may be desirable to include connection resource state in the link state information. Link metric changes could introduce sudden change in the load distribution. Achieving graceful changes in load distribution upon link metric update would be an important question to study further.

As a compromise between static and adaptive routing, the quasi-static routing can be considered with several metric sets for different time-of-day or day-of-week traffic pattern. In that case also, connection resource notion is important and the graceful change is required to reduce fluctuation or any instability.

We assumed even distribution of offered load on equi-distant shortest paths. The optimal solution's flow distribution is not necessarily even. To implement the uneven optimal distribution with SP routing, we need to have additional weight parameters. But it is very hard to provide the weights because every different path should have its own weight sets, i.e. scalability is poor. On the other hand, with the even distribution we cannot achieve the same level of routing performance as the optimal flow distribution. This is a basic trade-off between routing performance and implementation simplicity. The proposed algorithm attempts to help us determine optimal paths for minimum congestion with multi-resource notion and to generate link metric set to make the optimal paths to be equi-distant shortest paths.

Some routers and switches can support two different link metrics in the opposite direction on the same link. The others support only one link metric that is applied in both directions. By adding such conditions as $W_{ij} = W_{ji}$, $Z_{ij} = Z_{ji}$ and $X_{ij} = X_{ji}$ to the formulated optimization problem, we can support the case of a single metric for both directions as well with the proposed algorithm.

Note that the primal problem is close to infeasible because we are using tight constraints whence the dual problem may in practice appear to be unbounded. In our experiment this caused the solver easily generate 'solution unbounded' messages. Putting additional upper bound constraint to the dual variable $Z_{(i,j)}$ in (3.30) helped us to evade the difficulty in the experiments.

3.5 Conclusions

We presented the importance of connection resource notion in multi-service networks in order to achieve a well balanced bandwidth and signaling load and introduced the notion into the traffic engineering. We set up a network flow optimization problem which incorporates both the bandwidth and connection resource constraints. We proposed a new algorithm to compute the link metrics for SP routing which realizes the optimal paths for the formulated problem as the shortest paths. The result of our numerical experiments indicate the proposed algorithm is very effective. The algorithm can be implemented off-line and the generated single set of link metric has immediate applicability to connection-oriented framework such as PNNI or MPLS. Without any parameter addition to the current routing framework, by only assigning a new link metric set, the congestion cost which includes connection load as well as bandwidth load information can be substantially reduced. According to the availability of the traffic information, the proposed algorithm can be used in various ways as in time-of-day or time-of-week fashion. Future study might consider exploring on-line distributed implementation of these ideas.

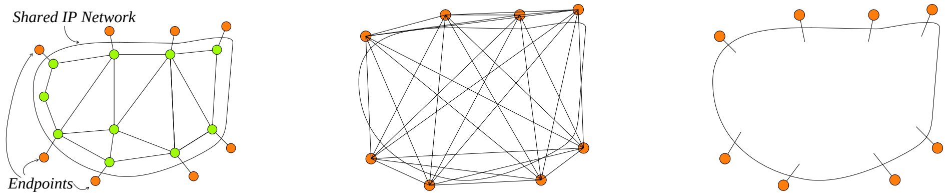
Chapter 4

Routing and provisioning VPNs on hose traffic models and/or constraints

4.1 Introduction

VPNs (Virtual Private Networks) are an approach to provide secure connectivity to geographically dispersed sites. They are virtual in the sense that they are overlayed on shared network resources. Meanwhile they are private in that only participating sites get access to the associated routing and address plan which is completely independent of the traffic on other VPNs sharing the same network[39]. In the best-effort context usually associated with the Internet, work on VPNs has for the most part focused on routing protocols [40] and security issues [41]. Secure connections are implemented using encapsulated IP payloads. Recent developments, towards supporting quality of service (QoS) for IP based traffic, i.e. MPLS [42] and Diffserv [43], suggest the use of VPNs as a means to support QoS, at a fairly 'coarse granularity', for a given distributed customer and/or application/service [44, 45].

The focus of our work is on using 'hose models' as a means to describe traffic demands on a virtual private network (VPN) and how traffic engineering, i.e. routing and provisioning, might be carried out in this context. By hose model, we refer to characterizations of the aggregate traffic in/out of a VPN's endpoints versus using a full origin-destination traffic matrix. This reduction in the required amount of information introduces uncertainty on what are the precise traffic loads on the network, which, for some cases, may make traffic engineering inefficient, while for others may enable higher efficiency and robustness to changing loads.



a) Sample network (b) Full traffic matrix (c) Hose model

Figure 4.1: VPN traffic characterization.

The 'hose model' was first introduced in the context of VPNs by [45], wherein four advantages were discussed: ease of specification, flexibility, multiplexing gain and characterization. Let us briefly summarize the arguments made therein. First, such a traffic characterization is simpler, see Figure 4.1, in that one need only specify aggregates at ingress/egress points, versus explicitly determining the demands between all origin destination pairs. This significantly reduces the amount of data required – data which in many cases one may be unable to measure/characterize reliably. Additionally a collection of hoses provides the customer with a natural service model. Secondly, provisioning based on hose models/constraints enables some flexibility in that the actual traffic demands offered to the network may vary as long as they are consistent with the hose characterization. That is to say, specifying traffic based on the hose model, and in turn dimensioning resources to meet these demands, facilitates dimensioning resources over a set of possible traffic matrices. Given the dynamic characteristics of traffic, one may indeed wish to allocate resources so that a set of possible traffic loads (matrices) can be supported. Unfortunately this flexibility may come at a resource cost. The third observation, is that provisioning resources based on hose specifications may allow additional multiplexing at the access points as well as within the network– this may alleviate the resource cost of the flexibility mentioned above. Finally, due to the smoothing resulting from aggregation at VPN endpoint's hoses, characterization of traffic may be facilitated. We note that the VPN hose traffic model also permits some flexibility with respect to dealing with multicast traffic loads, i.e., one need not account for multicasting explicitly, but only focus on what is seen at ingress/egress points.

Thus from the VPN service provider's perspective, provisioning a VPN based on the hose model can be advantageous – yet this is certainly not a simple task. Service providers have to solve two traffic engineering problems: (1) the selection of the paths (i.e. routing) to provide connectivity among end-points, and (2) the provisioning or resource allocation along those paths to meet the specified demands and possibly QoS guarantees for the traffic sharing the VPN.

Routing structures for VPNs. The initial work of Duffield et al.[45], and subsequent research, e.g., [46, 47], have significantly advanced our understanding of hose based VPN routing and provisioning, as well as the associated restoration problem [61]. As discussed in [45] one might consider routing VPN traffic in several ways:

- **Pipe Model:** a single route is determined (based on shortest path or other criteria) between each pair of points.

- **Ingress (Egress) Tree Model:** each ingress (or egress) of the VPN has a tree associated with it that carries traffic to (from) all egress (ingress) points. The tree might be routed based on shortest paths, or some other VPN or endpoint dependent metric. A VPN with n end points would be served by n possibly overlapping trees.

- **Shared Tree Model:** traffic between VPN end points is routed along a *common* tree. Various criteria might be used to route such trees, e.g., minimize the overall provisioning cost, use Steiner tree based on an appropriate link metric, or attempt to minimize congestion on the network.

- **Mesh:** an arbitrary routing of traffic, allowing the possibility of *splitting* traffic flows (e.g., for load balancing) among paths connecting VPN end points. If no splitting is permitted, a mesh reduces to a collection of individual pipes.

Depending on the routing structure, VPN dependent *routing state* may need to be maintained in the network, otherwise traffic would follow paths determined by the associated IP (or other standard) routing protocol. Indeed, explicit routing would be required to support routing structures specifically engineered for a given VPN. This can be achieved via source routing or by setting up and maintaining VPN dependent routing information in the network – e.g., maintaining appropriate

MPLS labels. Since overheads associated with such explicit routing are a concern, e.g., maintenance of MPLS labels requires non-negligible signaling among routers, shared trees are deemed to be advantageous [62].

Provisioning VPNs. To determine the resources that need to be *provisioned* [1] for a given routing structure, one needs to know whether VPN resource sharing state is available in the network and have a characterization/specification of the VPN's traffic. If the network supports VPN specific state for resource sharing, e.g., RSVP and MPLS labels, then pipes or trees that traverse a common link can share a capacity reservation on that link. Such sharing is particularly desirable when there are spatio-temporal variations in the VPN's traffic loads. The degree to which traffic load patterns are known, and the routing structure selected impact the extent to which such resource sharing can be exploited. For example, if a detailed traffic characterization is available (or measured) at each link, and sharing is supported, then resources might be appropriately provisioned to meet the aggregate VPN traffic demands on each link. By contrast if only hose descriptors are available, and no resource sharing is supported, provisioning is likely to be conservative and more costly. When only hose descriptors are available and resource sharing is supported for traffic on a shared VPN tree, an efficient provisioning of resources can be made, see [46, 47, 60]. In summary, various provisioning problems can be considered, depending on the routing structure, whether resource sharing is supported, and what is known about the VPN's traffic loads.

On dynamic resizing. Our discussion assumes that one is provisioning

---

[1] We do not distinguish here between explicit resource provisioning on behalf of a VPN and resources which are reserved through accounting mechanisms, and against which admission decisions would be made.

and possibly routing a VPN based on a prior characterization or constraints on the offered load. Such a characterization may be challenging to obtain due to spatio-temporal variations in the loads. For this reason, dynamic resizing of (possibly shared) VPN resources based on real-time (predictive) traffic measurements at network resources has been proposed [45]. Note however, that the additional efficiencies of resizing can only be reaped if other traffic is present to take advantage of the time-varying capacity being freed up. In order to admit new VPNs on the network, and guarantee reasonable QoS one will still need to make off-line allocations so as to ensure that surges in traffic can be met through resizing. Thus, the benefits of resizing depend on the bandwidth sharing/scheduling mechanisms being used across VPNs, the availability of 'best effort' traffic to use up the time varying capacity being made available, and the ability to perform aggressive VPN admission decisions. For example, if a work conserving service discipline is used to share resources across VPNs while ensuring a guaranteed rate to each, then resizing may not be worthwhile.

4.1.1 Contributions of this chapter

Previous work addressing the routing and provisioning problems was based on a simple worst case model for traffic at the VPN hose interfaces, i.e., a peak traffic on ingress and egress for each hose. As shown in [46, 47] in the symmetric case, i.e., when egress and ingress traffic loads are the same, the problem of selecting paths and provisioning the VPN can be transformed into a collection of shortest path problems from the end-points to various candidate roots, and where the provisioned capacity is additive, i.e., sum of the peak end point demands sharing shortest paths to the root.

In our work, we consider a richer set of traffic models for hose interfaces, e.g., characterized based on aggregate statistical properties, and ask how to route and provision a VPN to approximately meet desired QoS guarantees, e.g. call blocking or packet loss. The use of such statistical models can lead to additional cost savings by exploiting both spatial and temporal statistical multiplexing over VPN resources. We propose two useful hose traffic descriptors and a combination of them, and consider the path selection and provisioning problems that result. We show that accounting for the non-linearities resulting from temporal statistical multiplexing can provide some additional savings in the overall resource requirements of a VPN, particularly in the case where the end-points are neither sparse nor dense on the network of interest.

Attempting to minimize the provisioned or allocated resources to a given VPN customer may be of interest to customers if their costs are directly associated with those resources. However, in practice a VPN provider with multiple VPN customers may find that balancing the loads throughout a network is the more critical issue. Indeed if path selection and provisioning is managed so as to achieve a good balancing of loads, the VPN provider may be able to accommodate additional VPN customers. In particular the VPN provider may be able to cope with growth in the demand of current VPN customers and/or future additional new VPN customers, without requiring rerouting of currently configured VPNs. To adequately capture the characteristics of increases in the number of VPNs as well as demand growth, we consider three plausible models. The second contribution of this chapter is to investigate effective ways to deal with online VPN path selection and provisioning problem when new demands arise and rerouting is undesirable.

Additionally, we discuss some natural extensions of the hose service model, specifically the inclusion of multiple classes and cutset constraints, and consider how an application layer multicasting mechanism might balance loads over a VPN.

4.1.2 Summary of related work

The VPN hose model was first proposed in [45]. The provisioning issues on VPN hose model were studied by [46, 47]. Their work assumes the traffic is characterized based on the worst case or peak rate ingress/egress for each hose interface. Symmetric and asymmetric cases of the traffic demand were covered. The optimal and approximate algorithms were provided.

Restoration algorithm for VPN shared tree was studied in [61]; and for the reduction of label set for forwarding on shared VPN tree, label stacking issue was studied in [62].

In typical network flow problems, cost is modelled as a linear or convex function of the traffic load. However there are cases when concave functions are more realistic, e.g. applications where marginal cost decreases as load increases. The case where there are economies of scale due to statistical multiplexing considered in our work is one such example but concave costs appear in many situations in engineering and management[53, 54].

For typical network flow problems the set of feasible flows is convex. In this case extreme flows are defined as a feasible flows which cannot be represented by convex sum of other feasible flows. When the objective function being optimized is concave, it can be shown to have a finite global minimum on the feasible region, and there must be an extreme flow which is an optimal flow [52, 53]. Therefore if the number of extreme flows is finite, a search over extreme flows determines the optimal solution. But the enumeration may be laborious. In fact in general network flow problem with concave cost functions are known to be NP-hard[50]. Thus mixed integer programming and branch and bound method are the usual approach to tackle such problems[50, 51].

For network problems with special structure, finding the optimal solution may become manageable. For example, if the cost at all edges is linear except for one edge with concave cost, problem can be shown to be in class P[55]. Also for an acyclic single commodity single source multiple destination network flow problem, a dynamic programming based algorithm can be used[53].

For more general problems, however, it is difficult to find optimal solutions. To find feasible solution which is close to optimal solution, an alternative approach is to devise heuristic algorithms. In [56], an efficient heuristic approach for solving concave piecewise linear network flow problems was studied.

4.1.3 Organization of chapter

The rest of the chapter is organized as follows. In the next section, we discuss hose traffic descriptors and possible link provisioning functions for VPNs tree is presented. In Section 4.3, a framework to deal with provisioning a single VPN with concave cost functions on its edges is presented. In Section 4.4, we consider the multi-VPN provisioning problem to balance current traffic loads and future growth throughout a network. And then we discuss several further issues in Section 4.5.

4.2 Hose traffic descriptors and link provisioning functions for VPN trees

As first discussed in [45] one might consider routing VPN traffic in several ways: pipe model, ingress(egress) tree model, shared tree model, and mesh[60]. Pipe and shared tree structure are shown in Figure 4.2.

The shared tree structure exhibits nice properties in terms of reducing the state in the network, facilitating re-routing upon failure, and enabling spatio-temporal multiplexing on network resources. Spatially variable paths can be aggregated easily on tree structure with possible cost savings from spatial and temporal multiplexing. We will focus on the shared tree structure in this study.

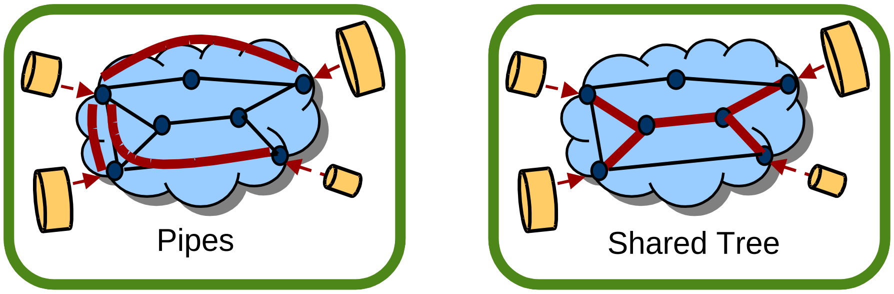


Figure 4.2: VPN routing structures.

We shall use the following notation to discuss provisioning and routing of shared VPN trees. The network is modelled as a graph $G(N, L)$ with a set of nodes N and a set of *bidirectional* links $L \subset N \times N$. A VPN has an associated set of end-points $E \subset N$ which are to be interconnected by a *shared tree* T on the graph – the tree is specified by a set of bidirectional links $T \subset L$. The removal of any link $(n, m) = l \in T$ partitions the tree into two sub-trees and thus the set of end-points into two sets L_l and R_l respectively. Finally we let $c_{n,m}$ and $c_{m,n}$ denote the capacity provisioned on link $l = (n, m)$ in the directions n to m and m to n respectively. Our notation is summarized in Table 4.1.

Table 4.1: Notation table for VPN hose model.

Notation	Description
G	Graph $G(N, L)$
N	Set of Nodes
L	Set of Bidirectional Links $N \times N$
T	Shared VPN Tree, s.t. $T \subset L$
E	Set of VPN Endpoints, s.t. $E \subset N$
L_l, R_l	Two sub-trees partitioned of a Tree T, s.t. $T \setminus l$, $l \in T$
$c_{n,m}$	Capacity provisioned on link $l = (m, n)$ in the direction n to m
$c_{m,n}$	Capacity provisioned on link $l = (m, n)$ in the direction m to n
D	$(D_{i,j} : i, j \in E)$, full (possibly unknown) traffic load matrix for VPN
H_i^{in}	Aggregate incoming (i.e. egress) traffic loads at $i \in E$
H_i^{out}	Aggregate outgoing (i.e. ingress) traffic loads at $i \in E$
$\phi_T(h)$	Total provisioning cost for T for hose vector h
$\rho_{n,m}(h)$	Peak (or mean) offered load on $(n, m) = l$ from n to m by hose vector h
x_{ij}^{tv}	Indicator(0/1) showing unidirectional edge (i,j) is in path (t,v)

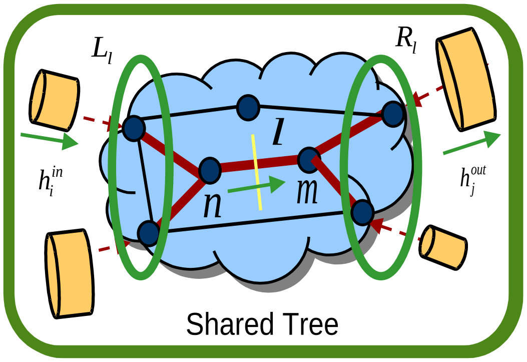


Figure 4.3: Provisioning shared trees.

We let $D = (D_{i,j} \ : \ i, j \in E)$ denote the full (possibly unknown) traffic load matrix for the VPN, where $D_{i,j}$ denotes the demand from VPN end-point i to j. One can determine 'hose' vectors H^{in}, H^{out} characterizing the *aggregate* incoming and outgoing traffic loads into the network at each of the VPN's endpoints, as

$H^{in} = Dv^T$ and $H^{out} = vD$, where v denotes a row vector with all 1's. More explicitly we have that

$$H^{in} = (H_i^{in} \,:\, H_i^{in} = \sum_{j \in E} D_{i,j},\; i \in E) \;\text{ and }\; H^{out} = (H_j^{out} \,:\, H_j^{out} = \sum_{i \in E} D_{i,j},\; j \in E).$$

We will focus on the fixed case where the demand matrix D is deterministic and thus so are the associated hose vectors $H = (H^{in}, H^{out})$. Note however that these could correspond to random matrices loads, see [60].

Suppose VPN loads D correspond to a single fixed matrix d, e.g., corresponding to the *peak (or mean)* point-to-point loads on the network. These might be measured in Erlangs for voice traffic or Mbps for data traffic. In this case the associated egress/ingress hose vectors, denoted by h^{in}, h^{out}, capture the aggregate peak or average offered traffic loads at VPN end-points assuming no losses/blocking. Note that while each matrix d results in a unique pair of hose vectors $h = (h^{in}, h^{out})$, the converse is not true. Thus if a VPN tree is provisioned based on h alone, there would be sufficient resources to support a family of matrices corresponding to all traffic splitting consistent with hose specifications at the endpoints. The basic idea for determining the capacity required on a given link of a shared VPN tree T is to determine the peak (or mean) offered load it could see. Recall that any bidirectional link $(n, m) = l \in T$ partitions the end points into two sets L_l and R_l. Assuming there are no losses, the peak (or mean) offered load $\rho_{n,m}$ across the link, from n to m, is given by

$$\rho_{n,m}(h) = \min[\sum_{i \in L_l} h_i^{in}, \sum_{j \in R_l} h_j^{out}], \tag{4.1}$$

with a similar expression for the offered load $\rho_{m,n}$ in the reverse direction. Eq. 4.1 corresponds to the minimum of the maximum incoming load from endpoints in L_l and the maximum load endpoints in R_l could sink. Depending on the application, the capacity required at this link might be $\rho_{n,m}(h) + \rho_{m,n}(h)$, or might correspond to a function of $c_{n,m}(\cdot)$ of the offered loads in each direction. These functions would typically be concave reflecting the economies of scale achieved by increasing the loads on a given resource. Thus the total provisioning cost $\phi_T(h)$ for a shared VPN tree T with hose load vectors h would be given by

$$\phi_T(h) = \text{ProvisioningCost}(T, h) = \sum_{l=(n,m)\in T} c_{n,m}(\rho_{n,m}(h)) + c_{m,n}(\rho_{m,n}(h)). \quad (4.2)$$

We note that for concave link provisioning functions the overall function ϕ_T is concave. If $c_{n,m}$ are linear then ϕ_T is radially homogenous, i.e., $\phi_T(\alpha h) = \alpha \phi_T(h)$. That is, a scaling of the hose vector results in a scaling of the provisioning cost.

Below we propose a few of possible traffic descriptors to be used to characterize the aggregate ingress/egress traffic at a hose interface. In each case we provide a simple provisioning function mapping the aggregate load and a QoS requirement to an estimate for the resources that must be allocated on a link. To simplify the provisioning one might simply ensure that the QoS at each link (e.g. call blocking probability for voice applications) is acceptable.

Our model includes the role of economies of scale and thus there is potential for cost savings if VPNs are routed so as to achieve high degrees of traffic aggregation on network resources even at the expense of taking longer paths.

4.2.1 Call level QoS - voice application

Suppose the VPN is to be used to support VoIP (Voice over IP) traffic. In this case each hose interface might be characterized by a symmetrical load in Erlangs, i.e., call arrivals per mean holding time. Suppose that calls require a fixed amount of bandwidth as in conventional telephony without loss of generality we assume this requirement is one unit and so a bidirectional link with capacity C units can support C concurrent voice connections without degradation in voice quality. As a QoS objective in routing and provisioning the VPN we focus on ensuring that the overall blocking probability is on the order of a pre-specified requirement δ. Although more sophisticated models capturing the interactions among demands between different end-points and links are possible [63], we will simply focus on meeting this requirement on each link of the VPN. If connection requests arrive as Poisson process, the blocking probability is given by Erlang's loss formula $E(\rho, C)$ where ρ is offered traffic load and C is the capacity and where

$$E(\rho, C) = \frac{\frac{\rho^C}{C!}}{\sum_{j=0}^{C} \frac{\rho^j}{j\,!}}. \tag{4.3}$$

In order to meet the call blocking QoS requirement of δ on a link supporting a load of ρ, one needs to invert this function to find the capacity that must be provisioned, i.e., determine the inverse Erlang formula $C(\rho, \delta) = E^{-1}(\delta, \rho)$. This can be computed in many ways depending on the regime of interest, see e.g., [49]. For a fixed δ one can show that the inverse Erlang function is increasing and strictly concave in ρ[49]. Thus, it is subadditive, i.e.,

$$C(\rho_1 + \rho_2, \delta) < C(\rho_1, \delta) + C(\rho_2, \delta). \tag{4.4}$$

Given a fixed QoS requirement δ the function $C(\rho, \delta)$ gives the bandwidth that must be provisioned as a function of the load ρ. Due to its concavity, as the load ρ increases, the marginal increase in required resources decreases, whence one can achieve a cost savings when aggregating traffic on common links. Alternatively, as exhibited in (4.4), assigning two demands with intensity ρ_1 and ρ_2 on separate resources incurs more cost than aggregating them in a single link. Fig.4.4 shows provisioning cost $C(\rho, \delta)$ (i.e. number of virtual trunks) versus offered load (denoted 'Rho') subject to three different blocking probabilities. As will be shown in the sequel the concavity of the provisioning function leads to certain natural structures for routing and provisioning VPNs.

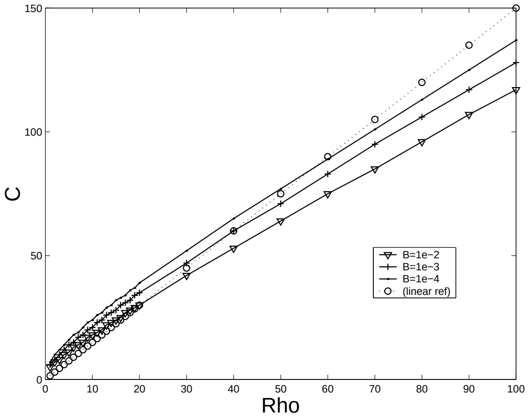


Figure 4.4: Provisioning cost - voice application.

4.2.2 Packet level QoS - bursty data traffic application

Suppose, that traffic at the VPN hose is associated with data traffic that has not been significantly aggregated and hence is potentially quite bursty, e.g., might be the aggregate of a collection of bursty users each of which is leaky bucket constrained. A simple model to capture the bursty characteristics of such traffic at the VPN end-points might be to specify the aggregate by an approximate number n of *homogeneous* independent on/off flows. Each flow i has a random rate X_i that has peak p during its on-period and a mean rate m. Hose traffic loads would thus be specified based on a number of such 'mini-sources'.

In order to provision a link shared by n such flows so as to ensure that the probability of loss is no more than δ one might determine C based on a simple bufferless model, i.e.,

$$P(\sum_{i=1}^{n} X_i > C) \leq \delta,$$

There are many ways to approximate the bandwidth $C(n, \delta)$ required to meet this QoS constraint. Perhaps the simplest is to approximate the offered load by a Gaussian distribution with the same mean and variance, i.e.,

$$\sum_{i=1}^{n} X_i \sim N(nm, nm(p-m)).$$

With this approximation the required bandwidth becomes

$$C(n, \delta) = nm + t(\delta)\sqrt{nm(p-m)}.$$

The parameter $t(\delta)$ is determined according to the QoS δ. Again $C(n, \delta)$ is concave in n for a fixed δ and thus exhibits economies of scale as traffic is aggregated on a given link.

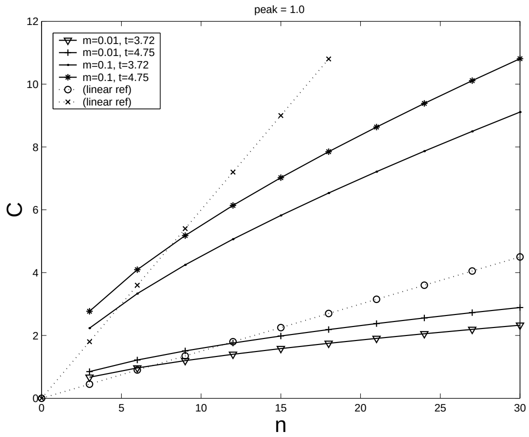


Figure 4.5: Provisioning cost - bursty data traffic.

We note that multiple extensions, e.g., to heterogeneous sources, [64] or more accurate provisioning approximations might be considered, yet perhaps in the case of VPN provisioning this may not be of great interest. A provisioning cost example plot is given in Fig.4.5.

4.2.3 Call and packet level QoS - voice application with silence suppression

The two models explained above can be merged together to produce another one. For example, consider a VPN intended to support IP telephony with silence suppression where the traffic loads at the hoses have been specified in Erlangs. Quality of service on such a VPN might be specified in terms of end-to-end call blocking probability δ_b and voice quality δ_v for successful calls. Thus to assure appropriate quality of service, sufficient resources would need to be allocated to meet the call blocking constraint, and admission control would need to be conducted to ensure voice quality. To simplify the provisioning problem one might simply ensure that the call blocking probability on each link is acceptable. The function $c_{n,m}(\cdot)$ in this case could be computed based on the inverse Erlang function, an appropriate allowable call blocking probability per link, and an appropriate packet level multiplexing model for voice connections with silence suppression. The bandwidth in the case might correspond to an *effective bandwidth* depending on acceptable packet loss characteristics [65]. This function would be concave and capture, on the one hand spatial multiplexing of resources on link (n, m), i.e. connections sharing that link but not the same path, and on the other hand temporal multiplexing of the resources by active connections that have silence suppression.

4.3 Routing VPN trees - minimizing provisioning cost

Let us first consider the VPN tree routing problem in the case where the objective is to minimize the total resources allocated or provisioned, i.e., cost of the VPN, and overall load balancing across the network is not a concern - this is the same problem considered by [46, 47]. We saw in the previous section that the economies of scale in the model have the potential of cost savings when VPN is routed to achieve traffic aggregation on network resources. The point we would like to investigate is to what extent the economies of scale impact optimal routing. The perspective here, is to simply 'greedily' minimize the resource costs associated with a given VPN, which may be deemed beneficial to either the provider, or VPN customer if these costs are passed on.

4.3.1 Problem formulation

VPN shared tree routing problem is to find a tree that best fits for the objective while connecting endpoints where hose descriptors are specified. It is shown in the Figure 4.6.

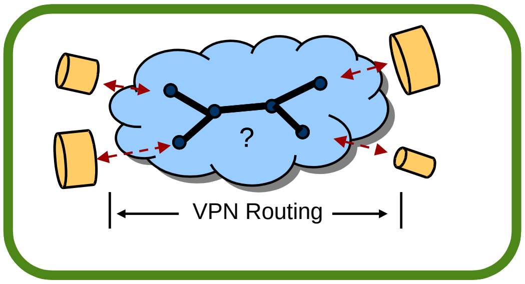


Figure 4.6: VPN shared tree routing problem.

We assume the traffic description at VPN endpoints are symmetric - i.e. the value of ingress and egress traffic descriptors are the same. For voice applications, this would of course be the case. Also when asymmetry is not too severe, one might take larger value out of egress and ingress traffic as representative of the traffic demand and the routing framework discussed in the sequel could be applied.

Note that asymmetric case problem with linear provisioning cost function was studied in [46, 47]. The problem turns out to be NP-hard even in the uncapacitated cases. To consider further concave provisioning cost cases, we may also need to consider heuristic approaches as in [46, 47].

We will use the general notation introduced in Table 4.1. If the bidirectional link (n, m) is selected to be part of VPN solution tree, there are two associated provisioned capacities $c_{n,m}$ and $c_{m,n}$ in opposite direction. The VPN tree routing and provisioning problem corresponds to finding a set of paths $\{P_{st} : s, t \in E\}$ which constructs a shared tree T and provisioned capacities $\{c_{n,m} : (n, m) \in T\}$. Here $c_{n,m}$ should be large enough to support the traffic intensity of all the feasible loads (n, m) might see subject to the VPN hose descriptor at the VPN endpoints. In other words, the provisioned capacity should support all the feasible loads given a constraint δ on the QoS. The capacity $c_{n,m}$ is set to zero if the link $l = (n, m)$ is not included in the tree.

The selection of paths should be done in a way to minimize the total capacity allocated, i.e. the provisioning cost. Suppose a shared tree T is given. The worst case feasible traffic intensity $\rho_{n,m}$ on the link (n, m) in T is given by Equation (4.1). By assuming symmetry in the hoses, i.e. $h_i = h_i^{in} = h_i^{out}$,

$$\rho_{n,m}(h) = \rho_{m,n}(h), \tag{4.5}$$

and by assuming symmetry in provisioning cost at each bidirectional link, the total provisioning cost is given by

$$\phi_T(h) = \text{ProvisioningCost}(T, h) = \sum_{l=(n,m)\in T} 2c_{n,m}(\rho_{n,m}(h)). \tag{4.6}$$

Therefore we pose the problem of routing a shared tree so as to minimize provisioning cost with a concave provisioning function as follows.

Problem 1 *Given a network $G(N, L)$, a set of endpoints $E \subset N$, and deterministic and symmetric hose descriptors $h^{in} = h^{out}$, to find a tree T connecting the endpoints E which minimizes the provisioning cost ProvisioningCost(T, h) (Eq. 4.6), where $c_{n,m}(\cdot)$ are concave.*

4.3.2 Problem characteristics and algorithm

Problem 1 is a complicated problem. We need to find a subset T of a graph G providing connectivity to the endpoints E in tree shape while minimizing the sum of the provisioned capacity at all links in T. Enumeration on all the possible trees in G might solve the problem but it is too laborious.

Directed tree introduced in [46, 47] is a useful tool in this context to study the characteristics of this problem. Suppose we have a feasible tree T. A directed tree T_d of T is given by assigning a direction to the links following the way described below.

By eliminating a link (n, m) in a tree T, we get two sub-trees $L_{n,m}$ and $R_{n,m}$ on n and m side respectively. Let $\rho(L_{n,m})$ and $\rho(R_{n,m})$ be the sum of demands at all the endpoints on the sub-tree $L_{n,m}$ and $R_{n,m}$ respectively. We obtain T_d by directing the edges (n, m), from the side that has bigger sum of VPN endpoint traffic demands to the side of smaller sum as follows: if $\rho(L_{n,m}) > \rho(R_{n,m})$, we put directed edge from n to m; if $\rho(L_{n,m}) < \rho(R_{n,m})$, we put directed edge from m to n; and if $\rho(L_{n,m}) = \rho(R_{n,m})$, we will arbitrarily direct the edge towards the side that contains a particular endpoint, say, $\hat{t}$. In this case we have the following fact.

Fact 1 *For any feasible tree T and directed tree T_d with whose edges have been directed according to the hose demands as discussed above, there exists a unique node $a(T_d)$ which has in-degree 0, and such that all the directed edges point away from $a(T_d)$. ([46, 47])*

By the Fact 1, once the node $a(T_d)$ is found, any link (n, m) on T_d is pointing from the side where $a(T_d)$ belongs to the other side. This gives a nice property that the feasible load on the link is given by the sum of traffic demand of endpoints on the side where $a(T_d)$ does not belong.

Here we can think of a network flow problem with multi-source single-sink single-commodity traffic. The idea is to set up a problem whose solution flows constitute a tree and are equal to the offered loads on the links in the directed tree T_d when the sink node v coincides with the node $a(T_d)$.

Let G be a graph with a set of nodes N and a set of bidirectional links $L \subset N \times N$. And $L' \subset N \times N$ is the unidirectional links derived from L. We denote a set of endpoints in N as E and h is the vector of hose descriptors h_e at $e \in E$. With the VPN endpoints $e \in E$ as sources with the supply amount h_e and a node v in N as a sink with the demand amount $\sum_{e \in E} h_e$, a network flow problem is precisely defined as follows. Since it comes down to nonlinear integer programming, we denote it as $NIP(G, v, h)$.

$NIP(G,v,h)$:

$$\min \sum_{(i,j)\in L'} 2\ \ c_{i,j}\big(\sum_{t\in E} x_{ij}^{tv}\ h_t\ \big)$$

subject to

$$\begin{aligned}
\sum_{j:(t,j)\in\ L'} x_{tj}^{tv} \ - \sum_{k:(k,t)\in\ L'} x_{kt}^{tv} &= \ \ 1 \quad t\in E \\
\sum_{j:(v,j)\in\ L'} x_{vj}^{tv} \ - \sum_{k:(k,v)\in\ L'} x_{kv}^{tv} &= -1 \quad t\in E \\
\sum_{j:(i,j)\in\ L'} x_{ij}^{tv} \ - \sum_{k:(k,i)\in\ L'} x_{ki}^{tv} &= \ \ 0 \\
t\in E\,,\ i\neq t\,,\ i\neq v\,,\ i\in N &
\end{aligned}$$

$$x_{ij}^{tv}\ \in\ \{\,0\ ,\ 1\}\qquad\qquad (i,j)\in L',\ \ t\in E$$

In $NIP(G,v,h)$ above, the integer variable x_{ij}^{tv} indicates whether a solution flow exists on edge (i,j) in the direction from i to j for the path (t,v).

The problem $NIP(G,v,h)$ corresponds to a network flow problem with concave cost. The flow conservation constraints and a no-splitting-of-flows constraint are given. Its solution exists, as will be an extreme flow, with solution paths constituting a tree structure [48].

Let the solution of $NIP(G,v,h)$ to be $\{x_{ij}^{*tv}\}$. We call the minimum cost in the objective function of $NIP(G,v,h)$ as *RootedCost*, namely,

$$\text{RootedCost}(G,v,h) = \sum_{(i,j)\in L'} 2\ \ c_{i,j}\big(\sum_{t\in E} x_{ij}^{*tv}\ h_t\ \big).$$

In the definition, x_{ij}^{*tv} is equal to 1 if edge (i,j) is on the solution path from t to the sink v with flow i to j. Otherwise, it is zero. We call the sink v the *root*. A tree T is defined to be feasible if $T \subseteq G$ and $E \subseteq T$. Then following key result holds.

Lemma 1 *Given a feasible tree T one can show that*

$$\mathit{ProvisioningCost}(T,h) = \mathit{RootedCost}(T,a(T),h). \tag{4.7}$$

Proof: By definition,

$$\text{ProvisioningCost}(T,h) = \sum_{(n,m)\in T} 2c_{n,m}(\min[\sum_{p\in L_{n,m}} h_p, \sum_{q\in R_{n,m}} h_q]) \tag{4.8}$$

and

$$\text{RootedCost}(T,a(T),h) = \sum_{(n,m)\in T'} 2\ \ c_{n,m}(\sum_{t\in E} x_{nm}^{*ta(T)}\ h_t\ ), \tag{4.9}$$

where T' denotes the tree of unidirectional links derived from T. If $(n,m)\in T$ then $(n,m)\in T'$ and $(m,n)\in T'$.

Let (i,j) be a link in T and suppose j is closer to $a(T)$ than i. And let $T_{i,j}^i$ be a subtree we get on i's side when we cut the link (i,j) in T. If we denote the solution of $NIP(T,a(T),h)$ by $\{x_{nm}^{*ta(T)}\}_{t\in E}$ on $(n,m)\in T'$, $\sum_{t\in E}\ x_{ij}^{*ta(T)}\ h_t$ stands for the solution flow on (i,j). Note that (i,j) is a cutset on the tree T and there are no sink but sources on $T_{i,j}^i$. By flow conservation, the solution flow on the cutset (i,j) is given by $\sum_{t\in E\bigcap T_{i,j}^i} h_t$. Note that $x_{ji}^{*ta(T)} = 0$ for all t. Therefore, in (4.9),

$$\sum_{t\in E} (x_{ij}^{*ta(T)} + x_{ji}^{*ta(T)})\ h_t = \sum_{t\in E\bigcap T_{i,j}^i} h_t. \tag{4.10}$$

We assumed the arrow on (i,j) points toward j on T_d. The definition of directed tree T_d gives the traffic intensity on (i,j) as

$$\begin{aligned}\rho_{i,j} &= \min[\sum_{p\in L_{i,j}} h_p, \sum_{q\in R_{i,j}} h_q] \\ &= \sum_{k\in T_{i,j}^i\bigcap E} h_k. \end{aligned} \tag{4.11}$$

By Equation (4.10) and (4.11), at each $(i,j) \in T$ where j is closer to $a(T)$,

$$c_{i,j}(\sum_{t \in E}(x_{ij}^{*ta(T)} + x_{ji}^{*ta(T)})) = c_{i,j}(\min[\sum_{i \in L_{i,j}} h_i, \sum_{j \in R_{i,j}} h_j]). \tag{4.12}$$

Similarly, at each (i,j) where i is closer to $a(T)$,

$$c_{i,j}(\sum_{t \in E}(x_{ji}^{*ta(T)} + x_{ij}^{*ta(T)})) = c_{i,j}(\min[\sum_{i \in L_{i,j}} h_i, \sum_{j \in R_{i,j}} h_j]). \tag{4.13}$$

In (4.8) and (4.9),

$$\begin{aligned} &\sum_{(n,m) \in T} 2c_{n,m}(\min[\sum_{p \in L_{n,m}} h_p, \sum_{q \in R_{n,m}} h_q]) \\ &= \sum_{(n,m) \in T} 2 \ c_{n,m}(\sum_{t \in E} x_{nm}^{*ta(T)} \ h_t \ + x_{mn}^{*ta(T)} \ h_t \ ) \\ &= \sum_{(n,m) \in T'} 2 \ c_{n,m}(\sum_{t \in E} x_{nm}^{*ta(T)} \ h_t \ ). \end{aligned}$$

Therefore (4.7) holds. (Q.E.D.)

Lemma 2 *Given a feasible tree T and node $r \in T$ one can show that*

$$\mathit{RootedCost}(T, a(T), h) \leq \mathit{RootedCost}(T, r, h)$$

i.e., among all candidate nodes for root of T, $a(T)$ gives the least RootedCost.

Proof:

We show the following property first: let (i,j) be a link in T and suppose j is closer to $a(T)$ than i. Then,

$$\text{RootedCost}(T, i, h) \geq \text{RootedCost}(T, j, h). \tag{4.14}$$

Let $T_{i,j}^{i}$ denote a subtree we get on i's side when we cut the link (i,j) in T. If we denote the solution of $NIP(T, i, h)$ by $\{x_{nm}^{*ti}\}_{t \in E}$ on $(n,m) \in T'$, with T' being the tree of unidirectional links derived from T, $\sum_{t \in E} \; x^{*ti}_{ji} \; h_t$ stands for the solution flow on $(j,i) \in T'$. Similarly, with solutions $\{\tilde{x}^{*ti}_{nm}\}_{t \in E}$ on $(n,m) \in T'$, $\sum_{t \in E} \; \tilde{x}^{*tj}_{ij} \; h_t$ stands for the solution flow of $NIP(T,j,h)$ on $(i,j) \in T'$.

By definition,

$$\text{RootedCost}(T,i,h) = \sum_{(n,m) \in T'} 2 \;\; c_{n,m}(\sum_{t \in E} \; x^{*ti}_{nm} \; h_t \;), \tag{4.15}$$

$$\text{RootedCost}(T,j,h) = \sum_{(n,m) \in T'} 2 \;\; c_{n,m}(\sum_{t \in E} \; \tilde{x}^{*tj}_{nm} \; h_t \;), \tag{4.16}$$

The link (i,j) is a cutset on the tree T and there are no sink but sources on $T^{j}_{i,j}$ for $NIP(T,i,h)$. By flow conservation, the solution flow of $NIP(T,i,h)$ on the cutset (i,j) is given by $\sum_{t \in E \bigcap T^{j}_{i,j}} h_t$. Note that $x^{*ta(T)}_{ij} = 0$ for all t. Therefore, in (4.15),

$$\sum_{t \in E} \; (x^{*ti}_{ij} + x^{*ti}_{ji}) \; h_t = \sum_{t \in E \bigcap T^{j}_{i,j}} \; h_t. \tag{4.17}$$

In similar way,

$$\sum_{t \in E} \; (\tilde{x}^{*tj}_{ij} + \tilde{x}^{*tj}_{ji}) \; h_t = \sum_{t \in E \bigcap T^{i}_{i,j}} \; h_t. \tag{4.18}$$

Note that, by flow conservation, all the other solution flows of $NIP(T,i,h)$ and $NIP(T,j,h)$ are the same, that is

$$\sum_{t \in E} \; x^{*ti}_{nm} \; h_t = \sum_{t \in E} \; \tilde{x}^{*tj}_{nm} \; h_t \quad (m,n) \in T', \;\; (m,n) \neq (i,j) \;\; and \;\; (m,n) \neq (j,i). \tag{4.19}$$

From the assumption that j is closer to $a(T)$ than i, following inequality holds.

$$\sum_{t \in E \bigcap T^{j}_{i,j}} h_t \;\geq\; \sum_{t \in E \bigcap T^{i}_{i,j}} h_t \tag{4.20}$$

By (4.17), (4.18), (4.19) and (4.20), Eq. (4.14) holds.

Now consider an arbitrary path from $a(T)$ to $r \in T$. By the Fact 1, all the arrows on the path in T_d points towards r. By applying the relation (4.14) iteratively, we prove the lemma. (Q.E.D.)

Theorem 1 *Suppose for every node in the graph, $r \in G$, we determine a feasible tree, T_r with minimal RootedCost, i.e.,*

$$T_r \in \arg\min_T \{\mathit{RootedCost}(T, r, h) | \mathit{feasible\ trees}\ \ T \in G, r \in G\}$$

Among all such trees, let T_{r^*} denote the one associated with node r^* which has minimal RootedCost, i.e.,

$$T_{r^*} \in \arg\min_r \{\mathit{RootedCost}(T_r, r, h) | r \in G\}$$

then

$$\begin{aligned} C(T_{r^*}) &= \mathit{RootedCost}(T_{r^*}, r^*, h) \\ &\leq \mathit{RootedCost}(T_{opt}, a(T_{opt}), h) = C(T_{opt}) \end{aligned}$$

thus T_{r^*} is a feasible tree with minimum provisioning cost, i.e., optimal.

Proof: Recall that T_{opt} denotes a feasible tree with minimal provisioning cost. It follows by Lemma 1 that

$$\text{RootedCost}(T_{opt}, a(T_{opt}), h) = C(T_{opt}).$$

Now since T_{r^*} is a tree with minimal RootedCost over all possible roots in the graph, it follows that

$$\text{RootedCost}(T_{r^*}, r^*, h) \leq \text{RootedCost}(T_{opt}, a(T_{opt}), h)$$

Finally since T_{r^*} is a tree with minimal RootedCost over all possible roots, it also follows by Lemma 2 that for any other node r

$$\text{RootedCost}(T_{r^*}, r^*, h) \le \text{RootedCost}(T_{r^*}, r, h)$$

whence $r^* = a(T_{r^*})$ and so $C(T_{r^*}) = \text{RootedCost}(T_{r^*}, r^*, h)$. It follows that T_{r^*} is a feasible tree with minimal provisioning cost. (Q.E.D.)

The above results essentially convert the problem of computing a minimal provisioning cost tree to the problem of computing a feasible tree with minimal rooted-cost for every root node in the graph. After solving $|N|$ $NIP(G, v, h)$ problems, one for each $v = r \in G$ as shown in Figure 4.7, we select the tree which has the minimal rooted-cost. By Theorem 1, it follows that this minimizes the provisioning-cost and hence the optimal shared tree for Problem 1 has been determined.

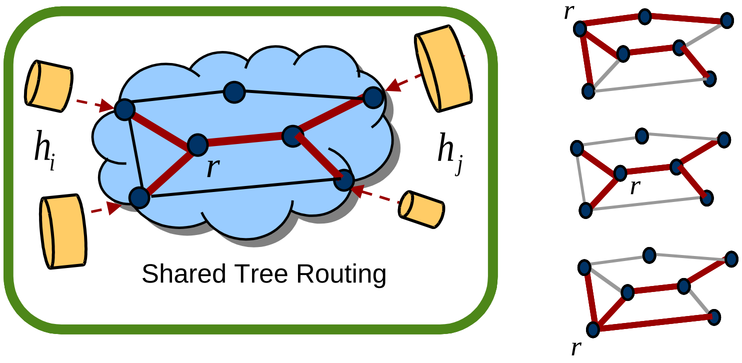


Figure 4.7: Routing shared trees - algorithm.

4.3.3 Heuristic algorithm

Although Theorem 1 provides an algorithm to find optimal shared tree for Problem 1, solving $NIP(G, v, h)$ is still an NP-hard problem because of the concave cost at the objective. If the objective is linear, solution is given by shortest path spanning tree which can be found by algorithms such as breadth-first search. We propose to use a simple greedy heuristic algorithm to obtain quick solution to this problem which appears to be adequate.

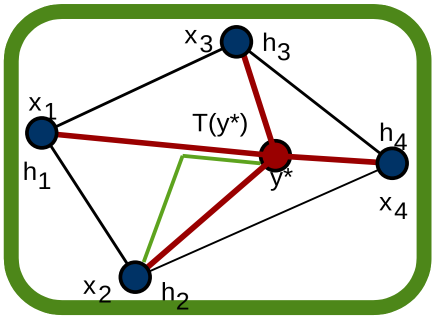


Figure 4.8: Routing shared trees - intuition.

Intuition behind our algorithm is sketched in the Figure 4.8. Given root y^* in the figure, the main objective of $NIP(G, y^*, h)$ is to find paths from the endpoints $\{x_i\}$ to y^*. Suppose the paths are set up in the increasing order of index i. Since the aggregation of paths gives the benefit of the economies of scale, if multiple number of equi-distant paths are available to the following endpoints, they are encouraged to follow all or part of the already established paths. Trade-off occurs when we need to decide whether to follow detouring path to utilize statistical multiplexing or to just follow shortest path, as shown for x_2 in the Figure 4.8.

We use shortest path algorithm to find path to root for each endpoints. We use the provisioning cost to set the link metric l_e for each link e. Before each path setup, we update the link metric to account for the statistical multiplexing provided by already established paths. More precisely, we use the marginal increase in the provisioning cost caused by the path setup of current endpoint with demand H_w.

The metric is given by

$$l_e = c_e(\rho_e + H_w) - c_e(\rho_e).$$

The current offered load on a link e is ρ_e. The demand at an endpoint w is H_w. Provisioning cost function $c_e(\cdot)$ calculates capacity to support the traffic. The link metric l_e is used in the shortest path calculation.

The order of path setup among endpoints is another issue. They may affect the performance of the utilization of statistical multiplexing. Regarding which endpoint to pick up first, our heuristic is to pick one that incurs minimum normalized cost increase to the network. For each path p^{wv} for endpoints w that are not yet connected, we calculate the following for each endpoint,

$$f[w] = \sum_{e \in p^{wv}} \frac{c_e(\rho_e + H_w) - c_e(\rho_e)}{H_w}.$$

We pick one that incurs the least normalized cost sum. The pseudo code of this heuristic is given in the Figure 4.9. We call it HEURCCVNFPSOLV() routine.

The node v is input as a root. The set variable Y contains the endpoints that are not connected to the shared tree yet. The current offered load on a link e is saved in ρ_e. The link metric l_e is used in the shortest path calculation routine $SP(G, w, v)$. The solution to the shortest path problem is implemented using common Djikstra algorithm. A shared tree T with provisioned capacity vector and total provisioning cost are returned.

The whole routine to find optimal shared tree is given in Figure 4.10. It numerates all nodes in the graph G for root and eventually returns minimum tree with the lowest provisioning cost.

```
**procedure** HEURCCVNFPSOLV($G$, $E$, $v$, $H$)
  $Y = E$
  $T = \phi$
  $C_e = 0,\ \rho_e^T = 0 \quad \forall e \in G$
  while ( $Y \neq \phi$ ) {
    $f_{min} = \infty$
    $w_{min} = y \in Y$
    $l_e = c_e(\rho_e + H_w) - c_e(\rho_e) \quad \forall e \in G$
    for each $w \in Y$ {
      $p^{wv} = \mathrm{SP}(G,w,v)$
      $f[w] = \sum_{e \in p^{wv}} (c_e(\rho_e + H_w) - c_e(\rho_e))/H_w$
      if ($f[w] \leq f_{min}$) {
        $f_{min} = f[w]$
        $w_{min} = w; \quad H_{w_{min}} = H_w$
      }
    }
    $T = T \cup p^{w_{min}v}$
    $Y = Y \setminus w_{min}$
    $\rho_e = \rho_e + H_{w_{min}} \quad \forall e \in p^{w_{min}v}$
    $\rho_e^T = \rho_e^T + H_{w_{min}} \quad \forall e \in p^{w_{min}v}$
  }
  $C_e = c_e(\rho_e^T) \quad \forall e \in T$
  $C_T = \{(e, C_e) | e \in T\}$
  $\phi_T = \sum_{e \in T} C_e$
  return ($T$,$C_T$,$\phi_T$)
```

Figure 4.9: Heuristic algorithm to solve NIP(G,v).

4.3.4 Results

In the experiment, we explore a case of voice application with silence suppression. Note that the traffic demand is symmetric in voice application. To model the provisioning cost with given traffic intensity in Erlang, we first get the number of connections to support connection-level QoS of blocking probability δ_c. Inverse Erlang function is used. We then use an on-off model (32 Kbps talk spurt period of average 0.4 sec and silence period of average 0.6 sec, see Easton et al.[67]) to support each connection with packet-level QoS of packet loss probability δ_p. We get statistical

```
procedure VPNRoutingProvCostSym($G$, $E$, $H$)
  $\hat{T}_{\hat{v}} = \emptyset$
  $\hat{v} = \emptyset$
  $\phi_{\hat{T}_{\hat{v}}} = \infty$
  $c_{\hat{T}_{\hat{v}}} = 0$
  for each node $v \in G$ {
    ( $T_v$, $C_{T_v}$, $\phi_{T_v}$) = HeurCcvNfpSolv ( $G$, $E$, $v$, $H$ )
    if ( $\phi_{T_v} < \phi_{\hat{T}_{\hat{v}}}$ ) {
      $\hat{v} = v$
      $\hat{T}_{\hat{v}} = T_v$
      $C_{\hat{T}_{\hat{v}}} = C_{T_v}$
      $\phi_{\hat{T}_{\hat{v}}} = \phi_{T_v}$
    }
  }
  Return ( $\hat{T}_{\hat{v}}$, $\hat{v}$, $C_{\hat{T}_{\hat{v}}}$, $\phi_{\hat{T}_{\hat{v}}}$ )
```

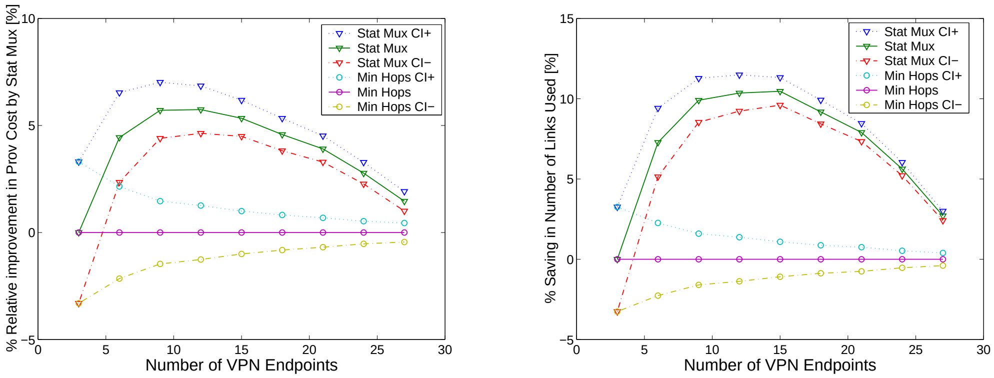
Figure 4.10: Algorithm to route near-optimal tree (symmetric case).

multiplexing gain from both inverse Erlang function and on-off source model.

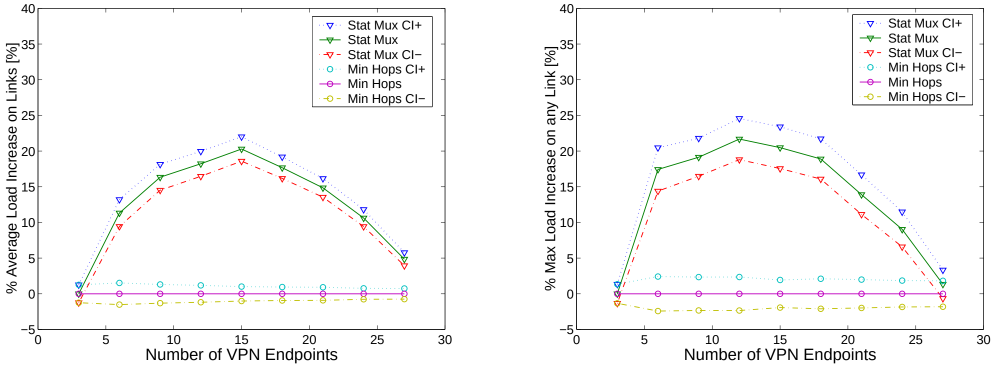
We compare our scheme that minimizes provisioning cost with the routing scheme that follows, in shared VPN tree setup, the minimum number of hops path from each endpoint to root. The latter routing scheme is the one proposed in [47, 46].

We use five different 30-node network topologies generated from Waxman model[68]. Average degree at each node is 3.4. Average Erlang traffic intensity at each VPN endpoint is assumed to be the same 1.0. As we vary the size of VPN in terms of the number of endpoint nodes included, we plot in Figure 4.11, (a) the percentage savings in total provisioning cost, (b) the percentage savings in the number of links used, (c) percentage increase in the average load in any link and (d) percentage increase in the maximum load in any link. The 95% confidence intervals are also shown in the plot.

The figures show that the cost savings achieved from a routing that is sensi-



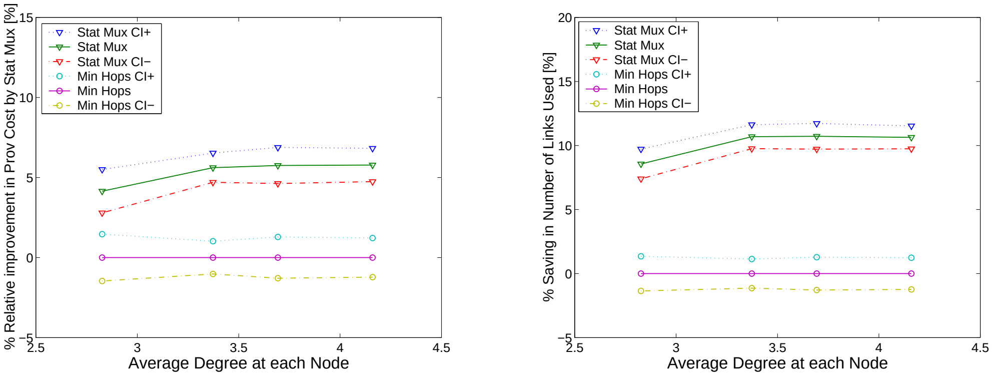
(a) Prov. Cost - Percentage Saving (b) Number of Links - Percentage Saving



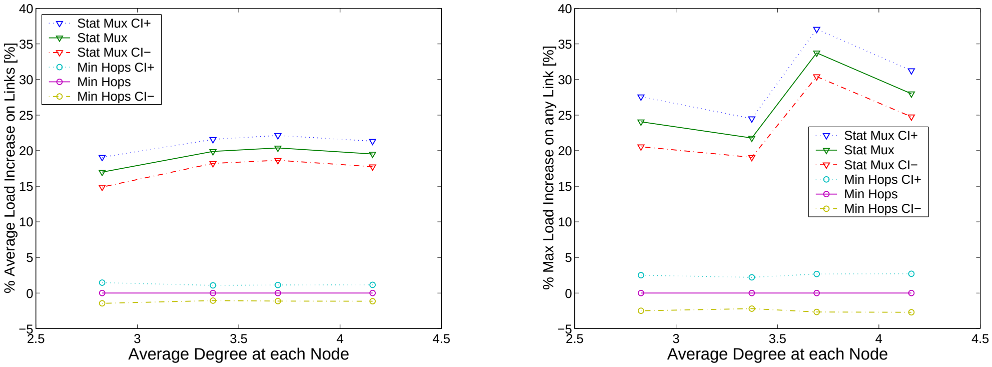
(c) Average Load - Percentage Increase (d) Max Load - Percentage Increase

Figure 4.11: Results versus number of endpoints with Waxman model topology.

tive to statistical multiplexing gains, are highest when the VPN includes a moderate number of nodes, versus a small or large numbers of nodes. In retrospect, this may be obvious, since VPNs with moderate numbers of nodes provide a significant number of opportunities for capacity sharing vs VPNs with small numbers of endpoints, yet this gain will not be overwhelmed by the need to include a large number of links to realize the necessary connectivity in large VPNs. Note that number of links used are also decreased when our algorithm is used. The reduced number of links means less maintenance. Since the traffic is aggregated in our algorithm, average load and



(a) Prov. Cost - Percentage Saving (b) Number of Links - Percentage Saving



(c) Average Load - Percentage Increase (d) Max Load - Percentage Increase

Figure 4.12: Results versus average node degree with Waxman model topology.

maximum load get increased.

The result plots with the same experiment setup except that we use network topologies generated from Pure Random model are attached in the Appendix B. Five different 30-node topologies are used and average degree at each node is 4. Plots show similar result pattern as the other case of different topologies.

In next experiment we vary the node degree. The plots are shown in Figure 4.12. The node degrees are in the range from 2.8 to 4.2. Five different 30-node topologies generated by Waxman model are used. We can see even though there is different topologies with different average node degree, the cost saving using our algorithm shows similar level of performance.

The plots of pure random graph model's case with same setup for varying node degrees are given in the Appendix B.

4.4 Routing VPN trees - enhancing network lifetime

4.4.1 Problem formulation

If the provisioning cost of a VPN translates to a service cost, then minimizing the provisioning cost may be desirable from the customer's point of view. However the provider may have other concerns. Indeed, a service provider will want to overlay various VPNs on his network. Thus a good VPN routing, from the provider's perspective, is one that allows as many of a sequence of VPN requests to be admitted. Moreover it would be preferable not to require re-routing, even if the customers subsequently increase their demands, i.e., renegotiate their hose parameters. As such one might ask how to route a sequence of shared-tree VPN requests so that the number of requests that these routing decisions can support is extended as long as possible. The VPN requests are in the form of either new VPN requests or demand increases in any of the current VPNs. We define this number of supported requests as *lifetime.* We use this lifetime as a performance metric to compare different routing schemes for VPN shared trees.

Online routing of connection requests is hard but well studied problem, see e.g., [66]. To see how this would work for routing shared trees, we propose two simple heuristics and make no effort to prove competitive optimality. In particular for each

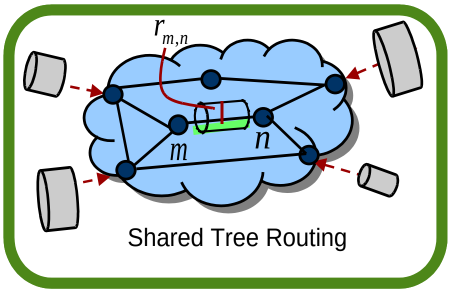


Figure 4.13: Congestion sensitive routing.

request to route a new VPN, we determine the minimum RootedCost tree over all possible roots in similar way as the provisioning cost minimization case that was discussed in the previous section. But this time the link costs are *convex* functions of the residual capacity on each link. Specifically we consider two such cost functions. The first is $1/r_{n,m}$ where $r_{n,m}$ denotes the capacity that has *not* been allocated on link (n, m) as shown in Figure 4.13. Thus finding trees that minimize the overall cost of this form will tend to avoid links that are congested. The second link cost function we consider is $v^{\alpha}_{n,m}/r_{n,m}$ where $v_{n,m}$ denotes the number of VPNs that are currently flowing through link (n, m) and $\alpha > 0$. The rationale for using this cost function is that a link with a high residual bandwidth but which already has a large number of VPNs going through it should be deemed congested, particularly when VPN demands are allowed to grow. Our experiments bear this out. See Figure 4.14 regarding future growth of VPN. Even though the residual capacity is the same in both cases, when VPN loads can grow, the load increase rate would be different in two cases.

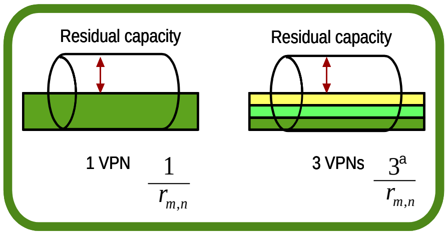


Figure 4.14: Congestion sensitive routing - future growth.

4.4.2 Heuristic algorithm

Note that the problem now has a convex cost as its objective and we are once again attempting to find the shared tree with minimum cost. The single commodity flow problem discussed earlier with a concave cost function no longer necessarily has a tree as its solution. In fact loops may occur in the optimal solution flows. Since we are interested in providing shared VPN tree, we introduce an additional step to the algorithm shown in Figure 4.15 which eliminates loops. It is denoted by $LoopElim(\cdot)$ and described in detail below.

The outline of the routing algorithm follows similar steps to those used in the cost minimization case shown in Figure 4.10. The inverse of available capacity R on a link is used as the link cost, namely $\frac{1}{R}$. Also another link cost function $\frac{n(VPNs)^\alpha}{R}$ is used. In this function, $n(VPNs)$ denotes the number of VPNs currently on a given link and $\alpha > 0$ impacts the sensitivity of the link metric to the number of VPNs on the link. Several values for α will be tested in the experiment.

The key pseudo code of the algorithm used to solve the network flow problem is shown in Figure 4.15. Given a root v and hose vector H, it returns a solution tree T. The set Y contains endpoints that have not been connected to the solution tree T. The set Y is initialized to the set of endpoints E. Each endpoint w in Y

```
    procedure HeurCvxNpfPrvs($G$, $W$, $v$, $H$)
1.    $Y = E$
2.    $T = \phi$
3.    $C_e = 0, \quad \rho_e^T = 0 \quad \forall e \in G$
4.    while ( $Y \neq \phi$ ) {
5.      $f_{min} = \infty$
6.      $w_{min} = y \in Y$
7.      $l_e = 1/(M_e - c_e(\rho_e)) \quad \forall e \in G$
8.      for each $w \in Y$ {
9.        $p^{wv} = \mathrm{SP}(G,w,v)$
10.       $p^{wv} = \mathrm{LoopElim}(p^{wv},T)$
11.       $f[w] = \sum_{e \in p^{wv}} (1/H_w)(1/(M_e - c_e(\rho_e) - H_w)$
12.                                $-1/(M_e - c_e(\rho_e)))$
13.       if ($f[w] \leq f_{min}$) {
14.         $f_{min} = f[w]$
15.         $w_{min} = w; \quad H_{w_{min}} = H_w$
16.       }
17.     }
18.     $T = T \cup p^{w_{min}v}$
19.     $Y = Y \setminus w_{min}$
20.     $\rho_e = \rho_e + H_{w_{min}} \quad \forall e \in p^{w_{min}v}$
21.     $\rho_e^T = \rho_e^T + H_{w_{min}} \quad \forall e \in p^{w_{min}v}$
22.   }
23.   $C_e = c_e(\rho_e^T) \quad \forall e \in T$
24.   $C_T = \{(e, C_e) | e \in T\}$
25.   $\phi_T = \sum_{e \in T} C_e$
26.   return ($T$,$C_T$,$\phi_T$)
```

Figure 4.15: Heuristic algorithm to solve NIP(G,v) for load balancing.

will be eliminated one-by-one after the selected path from w to the root v is used to add a path to the solution tree T. M_e denotes the capacity of link e whereas C_e stands for the provisioned capacity for the VPN under consideration. In the code, the link metrics are set according to the two costs described above on each iteration. In the Figure 4.15, the link metric update for the $1/R$ case is shown on line 7. The total traffic load by all the VPNs traversing the link e is denoted by ρ_e. The function $c_e(\cdot)$ denotes the provisioning cost function. For all the endpoints

w in Y, a standard shortest path solving routine $SP(G, w, v)$ is called on line 9. Among the paths that are determined p^{wv} from w's to v, some of the paths may result in loops in the tree T being established. The routine $LoopElim(p^{wv}, T)$ called on line 10 modifies the path p^{wv} so that, starting from w, once it reaches T, the path follows T to reach the root v. With this heuristic tree structure is preserved. Among these paths for w in Y, we select a path which incurs minimum congestion cost increase to the network. The cost increase is shown in the line 11 and 12. The selected path is connected to T on line 18 and the corresponding endpoint is then eliminated from the candidate set Y in the line 19. The current load ρ_e offered by all VPNs currently routed on the network which traverse on link e on the selected path is updated on line 20 by adding the selected endpoint's demand. Similarly, the load ρ_e^T for the VPN under consideration is updated on line 21. The iteration continues until the set becomes empty. The provisioned capacity C_e is found using a provisioning cost function $c_e(\cdot)$ in the line 23. The established solution tree T is returned with a provisioned capacity C_e at each link e on T. The total cost ϕ_T is also returned.

4.4.3 Network growth model

Requests to setup new VPNs as well as requests for growth in capacity of current VPNs both cause increases in load on the network. In our simulation experiments, we will generate a sequence of requests corresponding to either new VPNs or demand growth requests among current VPNs. At each simulation step we randomly decide whether it will correspond to a new VPN or growth in VPN load.

Each simulation step is indexed by an integer i, and we denote the time at which the i-th event (which is either a new VPN request or a VPN demand growth) arises by t_i.

Let $\hat{\gamma}(t_i)$ be the probability that new VPN is requested at time t_i. For simplicity we shall denote $\hat{\gamma}(t_i)$ by $\gamma(i)$, i.e., simply a function of the index i. We let $\gamma(i)$ be possibly time-varying function and given by

$$\frac{\lambda_{new}(t_i)}{\lambda_G(t_i) + \lambda_{new}(t_i)},$$

where $\lambda_{new}(t_i)$ is the current 'arrival rate' for new VPN requests and $\lambda_G(t_i)$ is the rate at which events corresponding to demand growths arrive to the network. The rate $\lambda_G(t_i)$ is given by $\sum_{v \in V} \lambda_{g,v}(t_i)$ where $\lambda_{g,v}(t_i)$ corresponds to the rate at which growth events occur for VPN v in the set V of current VPNs.

We will consider several models for the rate of requests to the system. Specifically we consider three scenarios. In each scenario, we assume that at each growth step the demand at each endpoint is increased by an additional random load which is uniformly distributed over $[5, 15]$ where 10 is the mean unit load. We let $x(t_i)$ denote the number of growths a VPN has seen prior to t_i. At each VPN growth step, we also increase $x(t_i)$ by the unit load 10. We call $x(t_i)$ the *load* of the VPN.

In this section we will provision capacity based on the peak demands at each link. Thus the load and provisioned capacity have a linear relation. If the load at each endpoint is doubled, we can show the provisioned capacity at each link on shared tree also gets doubled. Each new VPN request is assumed to have the constant number of endpoints.

Scenario 1: Suppose each VPN's load $x(t_i)$ grows linearly thus approximately $x(t_i) \simeq \lambda_g t_i$. New VPNs arrive at a fixed rate λ_{new}. Let $N(t_i)$ be the number of VPNs in the network at step i. The probability $\gamma(i)$ is given as

$$\gamma(i) = \frac{\lambda_{new}}{\lambda_g N(t_i) + \lambda_{new}}.$$

Using the normalization parameter $\beta = \lambda_{new}/\lambda_g$, $\gamma(i)$ is given by

$$\gamma(i) = \frac{\beta}{N(t_i) + \beta}.$$

If the selected event is not a new VPN, we need to select which VPN will grow among the current VPNs in the set V. In our simulations we did this uniformly, i.e., we select $v \in V$ with probability $\frac{1}{N(t)}$ and increase the load on VPN v by the unit load 10.

Scenario 2: Each VPN's load grows proportionally to its current bandwidth allocation, thus we have approximately $x(t_i) \simeq e^{\lambda_g t_i}$. New VPNs arrive at fixed rate λ_{new}. Let $x_v(t_i)$ be the load of a VPN $v \in V$ which grows by the unit load 10. Then,

$$\gamma(i) = \frac{\lambda_{new}}{\lambda_g \sum_{v \in V} x_v(t_i) + \lambda_{new}} = \frac{\beta}{\sum_{v \in V} x_v(t_i) + \beta}.$$

If the selected event is not of new VPN, we need to select a VPN to grow among the current set V. We select $v \in V$ with probability

$$\frac{x_v(t_i)}{\sum_{v \in V} x_v(t_i)}$$

and grow that VPN's load by a unit load. We can show this to lead to exponential growth in the VPN's load.

Scenario 3: In this case the load on each VPN will grow linearly and the VPN arrival rate is proportional to the number of VPN customers currently in the system. This results in exponential growth in demand for new VPNs as might be the case for a provider achieving market penetration. For this case we have

$$\gamma(i) = \frac{\lambda_{new} N(t_i)}{\lambda_g N(t_i) + \lambda_{new} N(t_i)} = \frac{\beta}{1+\beta} = \gamma_c.$$

These three types of scenarios will be used to load networks in our simulation experiments. Note that the parameter β has slightly different meaning in each scenario because the growth models are different. Still, it conveys how the new VPN arrival rate and demand growth rate are balanced. We investigate the performance of the proposed VPN shared tree routing algorithms using network lifetime as the performance metric.

4.4.4 Results

We conducted a fairly extensive set of experiments to evaluate routing of multiple shared VPNs. The first results are based on an 18-node real network topology (NSF net) taken from [69].

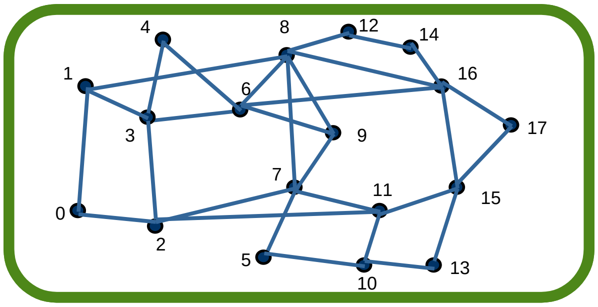
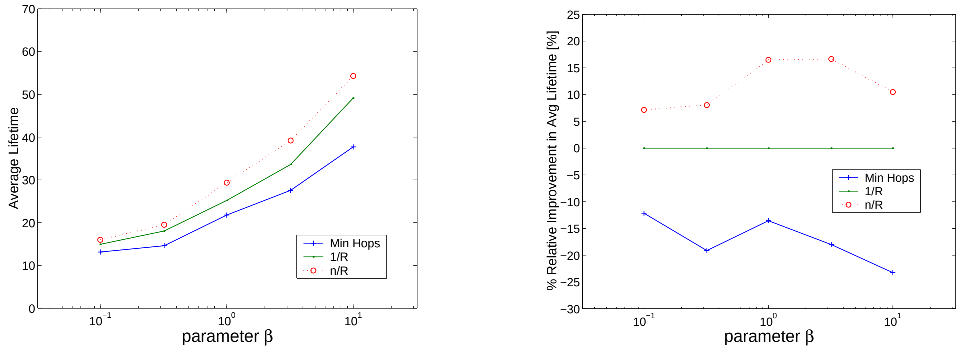


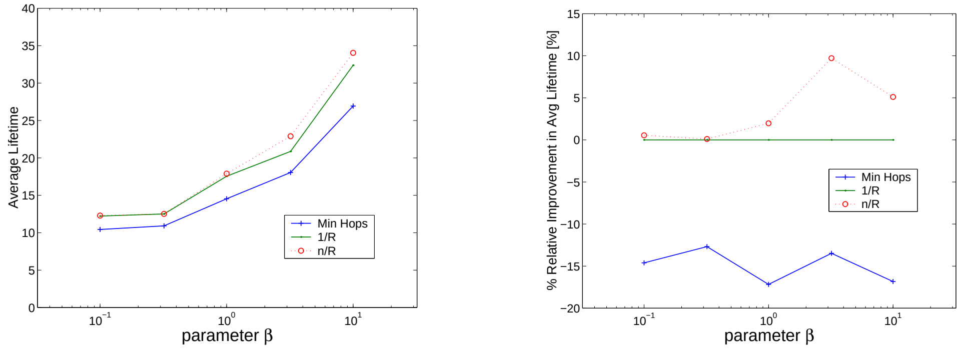
Figure 4.16: NSF topology.

In the experiment, we tested routing performance and varied the parameter $\beta = \lambda_{new}/\lambda_g$, the ratio between the arrival rate of new VPN versus the growth rate



(a) Average lifetime - Scenario 1

(b) Percentage increase in average lifetime - Scenario 1



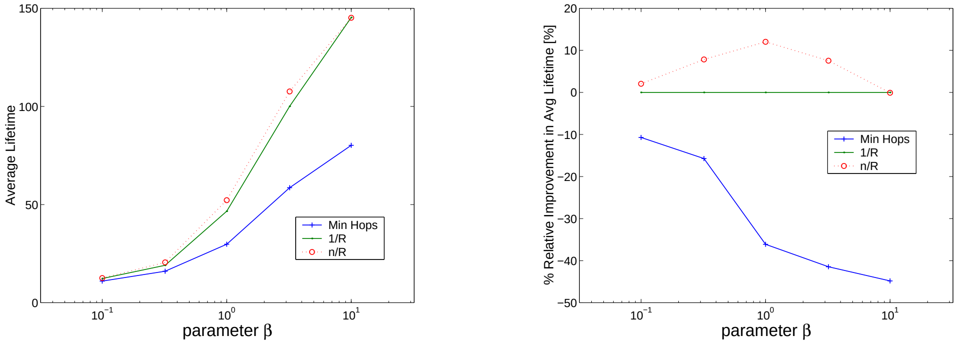
(c) Average lifetime - Scenario 2

(d) Percentage increase in average lifetime - Scenario 2

Figure 4.17: Results versus parameter β with real network topology (NSF net).

of a single VPN. We varied β from 0.1 to 10. For simplicity each VPN had five randomly selected endpoints, and loads were normalized so that the load associated with a growth was commensurate with the load associated with a new VPN. The provisioning of each link was done based on the peak of the load on the link. Three typical scenarios were all tested.

In the three scenarios, a routing scheme using the $n(VPNs)^{\alpha}/R$ metric with the number of VPNs n and residual capacity R shows good performance, when compared to $1/R$ as shown in Figure 4.17 and Figure 4.18. We also compared the



(e) Average lifetime - Scenario 3

(f) Percentage increase in average lifetime - Scenario 3

Figure 4.18: Results versus parameter β with real network topology (NSF net) - continued.

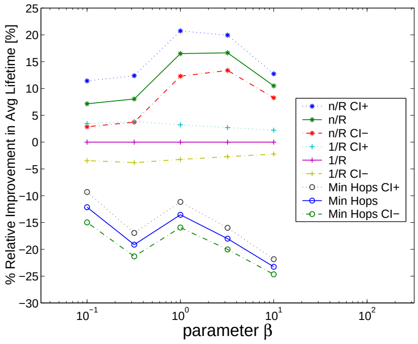
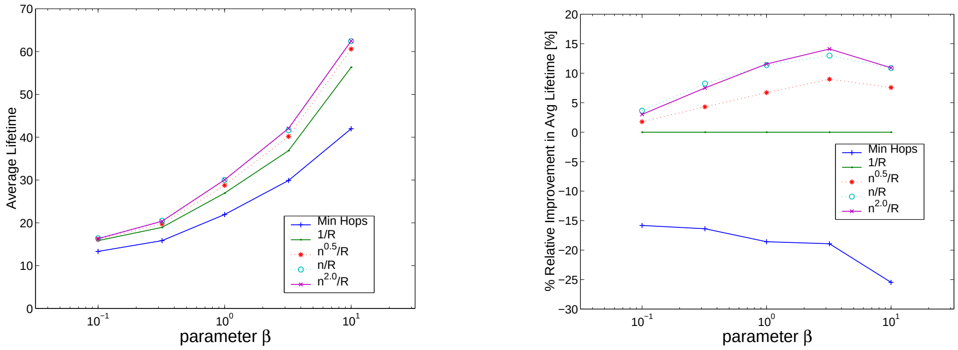


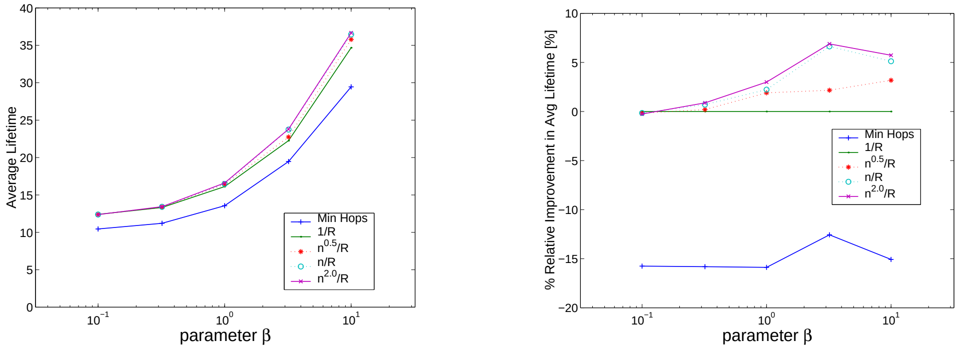
Figure 4.19: Confidence interval of percentage increase in average lifetime - Scenario 1 (NSF net).

routing scheme that follows, in shared VPN tree setup, the minimum number of hops path from each endpoint to root. The latter routing scheme is the one proposed in [47, 46]. The scheme is denoted by 'Min hops' in the plots. One representative 95% confidence interval of the results are shown in Figure 4.19. In these scenarios, as β gets bigger, i.e., when there are more new VPNs than growth requests, the network lifetime improves. This is because we do not allow rerouting of the already established VPN trees and new VPN can select routes that has available resources.



(a) Average lifetime - Scenario 1

(b) Percentage increase in average lifetime - Scenario 1

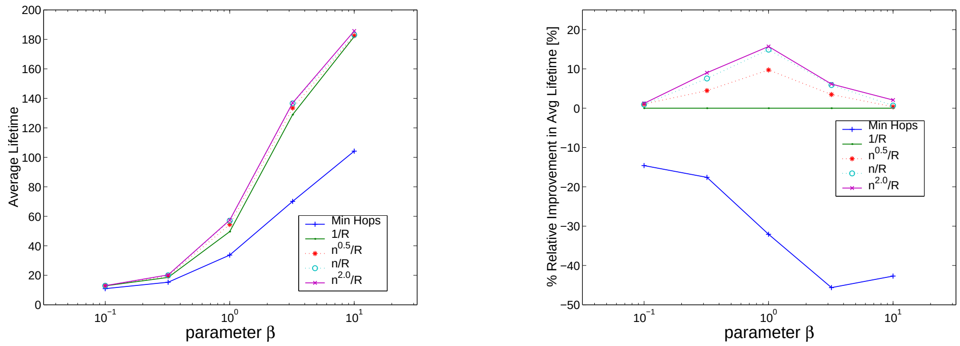


(c) Average lifetime - Scenario 2

(d) Percentage increase in average lifetime - Scenario 2

Figure 4.20: Results versus parameter β with Waxman model topology.

We observe that in Scenario 1 and 2, the maximum percentage increase in the lifetime is achieved when $\beta \simeq 10^{0.5}$. In Scenario 3, on the other hand, maximum percentage increase in the lifetime is found at more lower value of β around 1. Note that the β parameter has a different interpretation for each scenario. When β is one, only λ_{new} and λ_g are the same and profile of new VPN arrival and demand growth are different. We can interpret this difference in β value in each scenario when $\frac{n(VPNs)^\alpha}{R}$ based scheme shows good performance as the difference in the balance of the VPN-Growth and New-VPN characteristics. Note that among scenarios the VPN growth is the most severe in Scenario 2 since it grows exponentially. In Scenario 3, the total VPN growth and the total new VPN arrival characteristics are more balanced. So, the more intensive the VPN growth, the more new VPN arrival intensity is required to observe good performance based on the $\frac{n(VPNs)^\alpha}{R}$ routing metric. Therefore the maximum performance point gets drifted to the state of larger value of β.



(e) Average lifetime - Scenario 3

(f) Percentage increase in average lifetime - Scenario 3

Figure 4.21: Results versus parameter β with Waxman model topology - continued.

Figure 4.20 and Figure 4.21 show a second set of experiments where we use 5 different 30-node topologies for further study. One representative confidence interval of the results are shown in Figure 4.22. We generated the topologies using

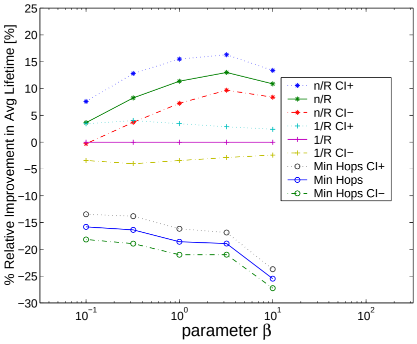


Figure 4.22: Confidence interval of percentage increase in average lifetime - Scenario 1 with Waxman model.

the Waxman model and generated an independent sequence of growth or new VPN arrivals until no more could be routed. Lifetime corresponds the total number of requests that are satisfied. Here we further consider the sensitivity of the number of VPNs in the routing metric with parameter α, i.e. the routing metric under test is n^{α}/R. We experimented with α set to 0.5, 1.0 and 2.0. Overall, n^{α}/R shows better performance than $1/R$ based and 'Min hops' routing in all three scenarios. Note that the results of n^{α}/R with different α values are insignificant in terms of confidence interval. In Figure 4.22, only the case of $\alpha = 1.0$ is shown but the 95% confidence intervals of other case with $\alpha = 0.5$ and 2.0 overlap with each other and mean values are included in all of the intervals. Thus, the routing performance is fairly insensitive to α. As in the case of real topology case, percent increase is maximum when $\beta = 1.0$ for scenario 3 but in scenario 1 and 2, the maximum percent lifetime increase comes around $\beta = 10^{0.5}$ where the new VPN request rate is bigger. Note that the overall and eventual growth still gets intensive than new VPN request arrival. Before it reaches the point, the network gets to the point of resource depletion.

We also did experiments using topologies generated by Pure Random model instead of Waxman model. The characteristics of results were similar and the scheme with $\frac{n(VPNs)^{\alpha}}{R}$ shows better performance over simple $1/R$ scheme. The plots included in the Appendix B.

4.5 Further issues

4.5.1 Extending the hose service model - cutset and multi-class constraints

In practice a vector h of hose descriptors might reflect contracts between the service provider and the VPN customer. A service contract specifies a distributed set of interfaces between the customer and the provider, i.e., how much traffic can enter and leave the network at each interface. Such contract would presumably be policed, and leading to excess traffic to be discarded or tagged for potential discarding elsewhere. The provider would presumably make a commitment to deliver all traffic conforming to the hose descriptors, and possibly delivering excess traffic if extra capacity is available in the network.

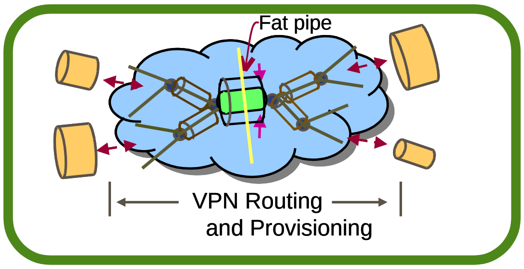


Figure 4.23: Extending the hose service model.

One problem with the shared tree solution is that it may require the allocation of a large amount of bandwidth in the 'center' of the tree. For example, in the worst case a link may need to be provisioned to meet about half the sum of the hose bandwidth agreements. For some VPNs this might be quite large, requiring the service provider to allocate a significant amount of bandwidth on a given resource, both to support the VPN and its restoration upon failure. A simple solution is to extend the service model to allow VPN customers or service providers to specify load constraints across specific (say transatlantic) links, each being taken as a 'virtual hose', and route the remaining traffic using the algorithms discussed previously.

Another useful extension to the VPN hose service model would be to allow specification of hose constraints for multiple traffic classes, i.e., a set of vectors $h^c, c \in C$. Once again, it may still be of interest to route the traffic on a common shared tree T. Provisioning T to satisfy multi-class constraints is a straightforward extension of Eq. 4.1 and 4.2, performed on a per class basis. However, determining a minimum cost tree to be *shared* by multiple classes appears to be difficult, yet it has some similarities with the asymmetric case considered in [46, 47]. One exception is the case where $c_{n,m}$ are linear and the hose vectors are colinear, in which case the minimum provisioning cost routing algorithm discussed above would be optimal for all classes.

4.5.2 Hose based VPN service model - a customer's perspective

In addition to providing peer-to-peer connectivity for various types of traffic among endpoints, VPNs might be used to multicast and cache large data files. If the VPN provider does not support network level multicasting, as is the case today, the VPN customer will need to resort to realizing multicasting himself. When this is the case, an attempt can be made to balance the load in a manner that is consistent with the endpoint hose parameters, and so as to minimize overall delays.

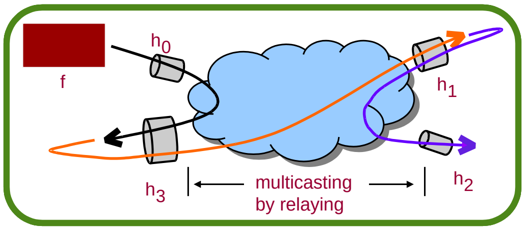


Figure 4.24: Large file to share with all endpoints.

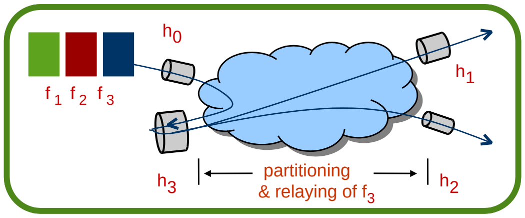


Figure 4.25: Application-level multicasting - load balancing

Consider a large file of size F to be distributed among a set of n endpoints $E = \{1, 2, \dots n\}$. Suppose without loss of generality that endpoint 1 is the originator of the file, and for simplicity that each endpoint $i \in E$ has a symmetric hose constraint $h_i = h_i^{in} = h_i^{out}$. A customer has several options to distribute such a file. The first is to transmit the file individually to each of the $n-1$ recipients. Since its hose constraint is h_1 in the best case, i.e., no other traffic in the network, this would requiring transmitting $(n-1)F$ units of data, for a duration of time $(n-1)F/h_1$ before *all* end points would approximately have received the file. In addition to the potentially high delay, this approach imposes a large and sustained load on endpoint 1's hose. Compare with the case in which the network supports network layer multicast. In this case only F data units would flow through the hose and the minimum time would be F/h_1. What other options does the application have?

One alterative is to resort to relaying the file or subfiles. The benefits here can be both in terms of distributing the ingress load across other endpoint hoses, which might be beneficial from a congestion or load balancing point of view, while also reducing delays. Consider the simple case where the file is forwarded from one endpoint to another, e.g., from 1 to 2, 2 to 3, until $n-1$ forwards the file to endpoint n. In this case each endpoint hose would see F units (both in and out), and the best case delay would be $F\sum_{i=1}^{n} 1/h_i$ if the file needs to be completely received prior to transmission. We shall refer to this as *forwarding.* Alternatively if data is *relayed* by endpoints as it is received, the best case delay would be $F/\min[h_i|i = 1, \dots n]$. Note that the ingress load to the VPN has been distributed equally among all hose endpoints. Note also that if hose constraints are equal, then the delay achieved by relaying is essentially that of network level multicasting. If there are unequal hose constraints then the overall transfer delay is dictated by the bottleneck hose.

A further alternative is to consider splitting the file into $n-1$ sub files, each with a size proportional to the hose constraint associated with the endpoint that will serve as its relay, i.e., the size of the subfile sent to hoses $i = 2, \dots n$ would be given by $f_i = \frac{h_i}{\sum_{j=2}^{n} hj}F$. Suppose further that the data originator initiates transmission of the subfiles to each endpoint concurrently and sends at rates proportional to their sizes, whence the reception rate to node i would be $r_i = \frac{h_i}{\sum_{j=2}^{n} hj}h_0$. In turn node i could relay the subfile to the remaining $n-2$ nodes through its hose with capacity h_i. Assuming full and fair use of the hose capacity the relay node i could achieve a rate of $h_i/(n-2)$ to each subsequent endpoint. If the condition

$$r_i = \frac{h_i}{\sum_{j=2}^{n} h_j} h_0 \leq \frac{h_i}{n-2} \quad \Rightarrow \quad h_0 \leq \frac{\sum_{j=2}^{n} h_j}{n-2}$$

is satisfied then the ideal overall delay would be F/h_0, i.e., that achieved by network level multicasting. In particular this would allow optimization of performance when there is heterogeneity in the hose parameters at each endpoint, yet one wishes to achieve the optimal performance allowable given the sender's endpoint. Note however that loads have been distributed in a different manner for this case, each relay node would see F units of data on its egress hose, and $2f_i$ units of data on its ingress hose – the congestion incurred by each endpoint is proportional to the hoses size, a reasonable criterion for balancing the loads on a shared VPN.

The key idea here is that once a customer has set up a VPN based on hose parameters, he may want to design applications which are sensitive to either the loads on the VPN or the service level agreements made with the provider. Thus for example, in the case of application level multicasting of data files the customer may have a degree of latitude to balance the loads on the VPN, so as to minimize interference with ongoing peer-to-peer traffic.

Chapter 5

Conclusion

In this thesis, we address three questions associated with traffic engineering and design of multi-service networks. A major concern throughout, has been scalability, i.e., how will the mechanisms used to manage congestion or design such networks scale as capacities, signalling loads, and numbers of users supported increase. In addition we have constrained the space of possible solutions, by focusing on mechanisms that represent an evolutionary path from current practice among service providers, or protocols that are in the process of being standardized.

In Chapter 2, we considered how to evolve flow control mechanism currently used and/or proposed for use in the next generation transport protocols for the Internet. We identified the problem of designing mechanisms that could function properly over a wide range of operational regimes. In particular we considered the effectiveness of feedback flow control combined with active queue management as the link capacity and/or the number flows grows. We found that mechanisms such as RED, could exhibit particularly poor performance when their parameters were not suitably tuned to the operational regime. To remedy this problem we proposed a simple modifications to both flow control and new active queue management mechanisms that exhibits robust performance, without requiring parameter tuning. We have validated our scheme through simulation, and showed its robustness and superior performance over RED.

In Chapter 3 of this thesis we discuss a novel routing problem currently faced by service providers. Indeed rather than adopting dynamic routing schemes that adapt well to the traffic loads, providers have opted to continue using shortest path routing based on as set of fixed link weights. In that light we consider how those weights might be set given a priori information on the traffic loads. This corresponds to solving an 'inverse' problem, i.e., determine a set of fixed weights for network links such that shortest path routing gives a 'good' flow distribution across the network. Our motivation was to determine how to set these weights so as to balance loads across two resource types: bandwidth and connection signaling resources in the network. Our main contribution, was to propose a systematic algorithm to determine such weights based on solving a two linear optimization problems. We have validated this algorithm, and showed that congestion costs on both resource types, are substantially reduced over traditional solutions.

In Chapter 4 we investigate various questions associated with routing and provisioning virtual private networks (VPNs) based on hose traffic models or constraints, i.e., using aggregate (hose) traffic characterizations in/out of VPN endpoints. This appears to be a promising simple approach to dealing with traffic engineering for VPNs. We evaluated the possible efficiencies derived from spatial multiplexing, i.e., over possibly varying routings of traffic loads consistent with hose specifications, and temporal multiplexing, i.e., statistical multiplexing of bursty loads, on shared VPN resources. In order to do so, we extended the state of the art in this area, by proposing an algorithm to route VPN shared trees which accounts for non-linear (concave) provisioning costs at network resources. This thesis includes a first study of online routing for VPNs requests, i.e., either new VPNs or growth in their associated demands, on a shared network. To this end we proposed a routing algorithm that significantly enhances the 'life-time' of routing decisions (i.e., to avoids re-routing) as well as the number of requests that can be accommodated. Finally we discuss some natural extensions of the hose service model, specifically, the inclusion of multiple classes and cutset constraints, and how application layer multicasting mechanisms might balance loads over a VPN's hoses.

The challenge posed in much of our work is to seek a middle ground between performance and complexity, while attempting to address problems in a manner that is consistent with standard practice. This has been difficult, yet exciting as traditional problems such as routing and provisioning come alive with utterly new formulations. Finding this middle ground may indeed be the key research challenge for the next decade of network research.

Appendix A

Addendum to Chapter 2

A.1 Linear program formulation for the piecewise linear congestion cost model

The optimization problem with piecewise linearized congestion cost function can be formulated to an LP problem as follows.

Minimize

$$\sum_{(i,j)\in E} \Big(\Phi_{(i,j),\,bw} + \Phi_{(i,j),\,cn} \Big)$$

subject to

$$\begin{array}{lll} L_{(i,j),\,bw} = & \sum_{(m,\,n)\in\,K} \beta^{(m,\,n)} X_{(i,j)}^{(m,\,n)} & (i,j) \in E \\ L_{(i,j),\,cn} = & \sum_{(m,\,n)\in\,K} X_{(i,j)}^{(m,\,n)} & (i,j) \in E \end{array}$$

$$\alpha_{bw} = \frac{C^*}{Max_{(i,j)\in E}(C_{(i,j),\,bw})}$$
$$\alpha_{cn} = \frac{C^*}{Max_{(i,j)\in E}(C_{(i,j),\,cn})}$$

$$\sum_{j:(m,j)\in\, E} X^{(m,n)}_{(m,j)} - \sum_{k:(k,m)\in\, E} X^{(m,n)}_{(k,m)} = \; b^{(m,n)} \quad (m,n)\in K$$

$$\sum_{j:(n,j)\in\, E} X^{(m,n)}_{(n,j)} - \sum_{k:(k,n)\in\, E} X^{(m,n)}_{(k,n)} = -\, b^{(m,n)} \quad (m,n)\in K$$

$$\sum_{j:(i,j)\in\, E} X^{(m,n)}_{(i,j)} - \sum_{k:(k,i)\in\, E} X^{(m,n)}_{(k,i)} = \; 0 \qquad (m,n)\in K\,,$$
$$i \neq m\,,\; i \neq n\,,\; i \in V$$

$$\begin{aligned}
\Phi_{(i,j),\,bw} &\geq 0 && (i,j)\in E\\
\Phi_{(i,j),\,bw} &\geq \alpha_{bw}(3L_{(i,j),\,bw} - \frac{2}{3}C_{(i,j),\,bw}) && (i,j)\in E\\
\Phi_{(i,j),\,bw} &\geq \alpha_{bw}(10L_{(i,j),\,bw} - \frac{16}{3}C_{(i,j),\,bw}) && (i,j)\in E\\
\Phi_{(i,j),\,bw} &\geq \alpha_{bw}(70L_{(i,j),\,bw} - \frac{178}{3}C_{(i,j),\,bw}) && (i,j)\in E\\
\Phi_{(i,j),\,bw} &\geq \alpha_{bw}(500L_{(i,j),\,bw} - \frac{1468}{3}C_{(i,j),\,bw}) && (i,j)\in E\\
\Phi_{(i,j),\,bw} &\geq \alpha_{bw}(5000L_{(i,j),\,bw} - \frac{16318}{3}C_{(i,j),\,bw}) && (i,j)\in E
\end{aligned}$$

$$\begin{aligned}
\Phi_{(i,j),\,cn} &\geq 0 && (i,j)\in E\\
\Phi_{(i,j),\,cn} &\geq \alpha_{cn}(3L_{(i,j),\,cn} - \frac{2}{3}C_{(i,j),\,cn}) && (i,j)\in E\\
\Phi_{(i,j),\,cn} &\geq \alpha_{cn}(10L_{(i,j),\,cn} - \frac{16}{3}C_{(i,j),\,cn}) && (i,j)\in E\\
\Phi_{(i,j),\,cn} &\geq \alpha_{cn}(70L_{(i,j),\,cn} - \frac{178}{3}C_{(i,j),\,cn}) && (i,j)\in E\\
\Phi_{(i,j),\,cn} &\geq \alpha_{cn}(500L_{(i,j),\,cn} - \frac{1468}{3}C_{(i,j),\,cn}) && (i,j)\in E\\
\Phi_{(i,j),\,cn} &\geq \alpha_{cn}(5000L_{(i,j),\,cn} - \frac{16318}{3}C_{(i,j),\,cn}) && (i,j)\in E
\end{aligned}$$

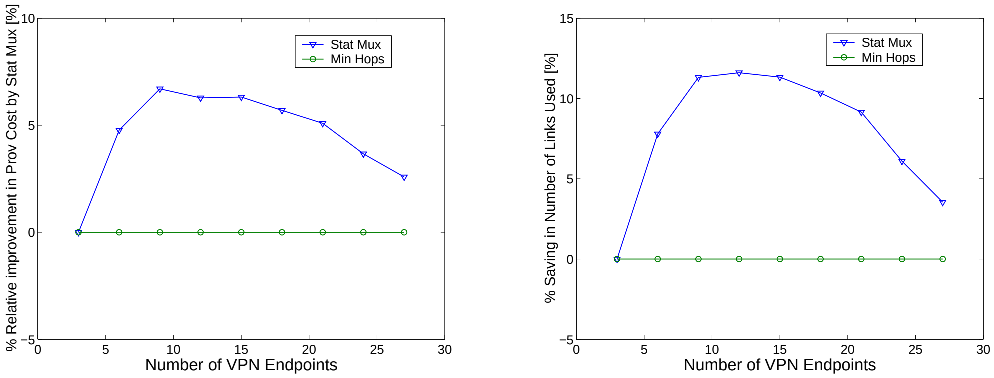
$$X_{(i,j)}^{(m,n)} \;\geq\; 0 \qquad\qquad (i,j)\in E\,,\;\; (m,n)\in K$$

Appendix B

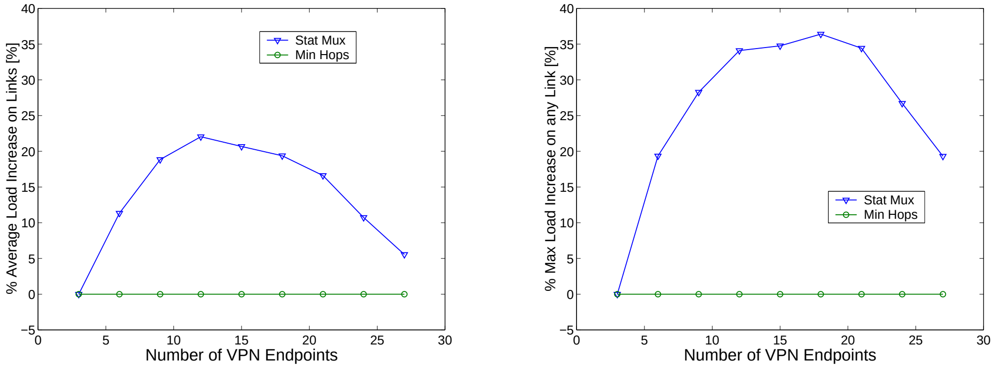
Addendum to Chapter 4

B.1 Minimizing provisioning cost routing experiment results - Pure Random model topology

The plots in Figure B.1 show representative results for experiments conducted on the provisioning cost minimization routing scheme. A voice application with silence suppression was considered. We use five different 30-node network topologies generated using a Pure Random model[68]. The average degree at each node is 4.0. The average Erlang traffic intensity at each VPN endpoint was assumed to be the same and equal to 1.0. As we vary the size of the VPNs in terms of the number of endpoint nodes included, Figure B.1 shows (a) the percentage savings in total provisioning cost, (b) the percentage savings in the number of links used, (c) percentage increase in the average load in any link and (d) percentage increase in the maximum load in any link. The confidence intervals, not plotted here, are similar to those in Figure 4.11.



(a) Prov. Cost - Percentage Saving (b) Number of Links - Percentage Saving

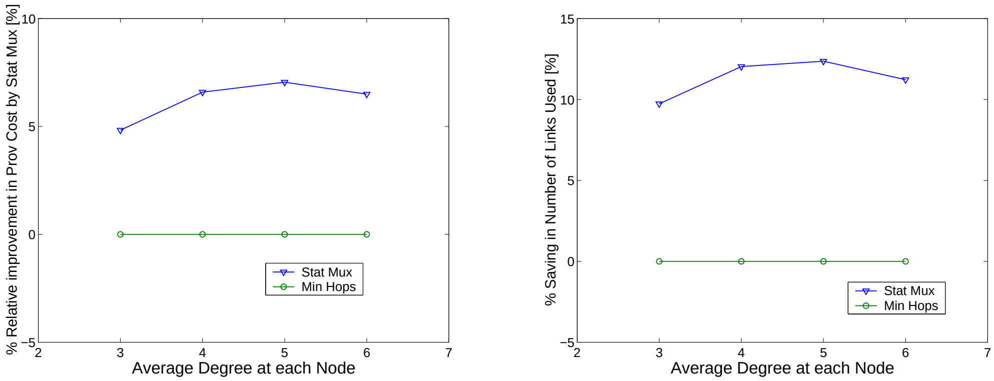


(c) Average Load - Percentage Saving (d) Max Load - Percentage Saving

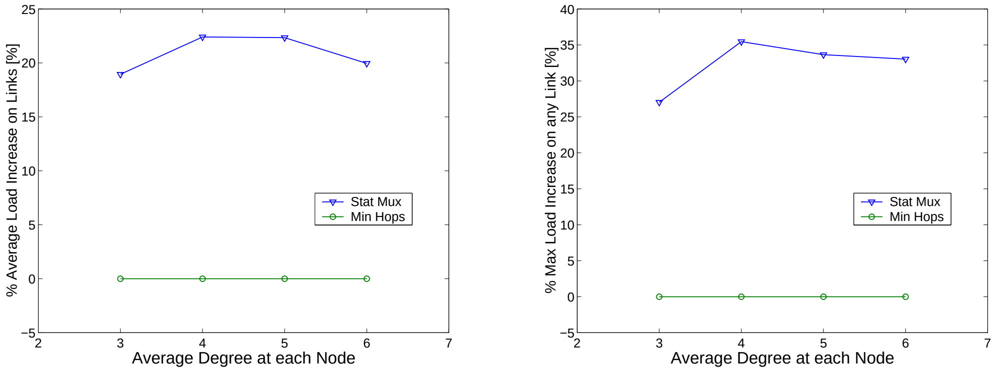
Figure B.1: Results versus number of endpoints with Pure Random model topology.

The figures show similar results to the case for network generated based on the Waxman. The cost savings achieved by making routing sensitive to statistical multiplexing gains, are highest when the VPN includes a moderate number of nodes, versus a small or large numbers of nodes. Our proposed scheme shows robust performance over different type of topologies.

In next experiment we vary the node degree of the network graph. The plots are shown in Figure B.2. The node degrees are in the range from 3.0 to 6.0. Five different 30-node topologies generated by Pure Random model are used.



(a) Prov. Cost - Percentage Saving (b) Number of Links - Percentage Saving



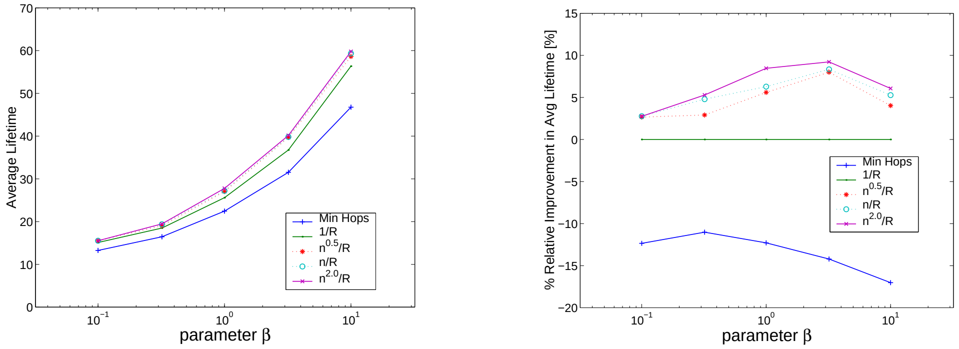
(c) Average Load - Percentage Saving (d) Max Load - Percentage Saving

Figure B.2: Results versus average node degree with Pure Random model topology.

As in the Waxman model based case, we can see even though there are different topologies with different average node degree, the cost saving using our algorithm shows performance that is insensitive to the difference.

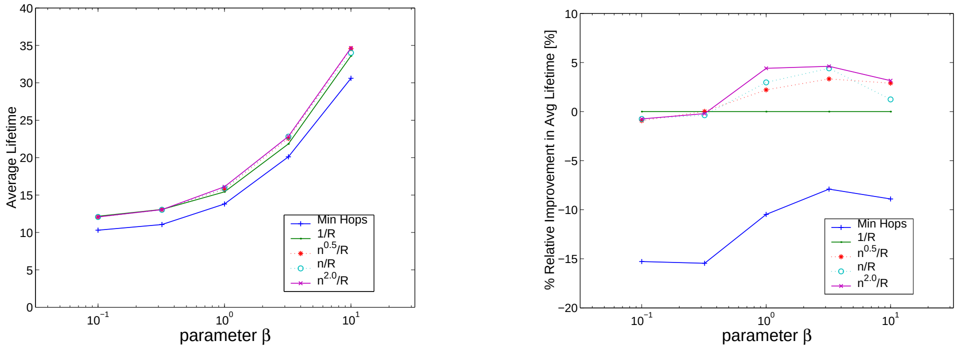
Figure B.3 shows a set of experiments where we use 5 different 30-node topologies to investigate the performance of the proposed routing algorithm maximizing network lifetime. One representative confidence intervals for these results are shown in Figure B.4. Here we generated the topologies using Pure Random model and generated an independent sequence of growth or new VPN arrivals until no more could be routed.

As in the real network and the Waxman model based cases, $n(VPNs)^{\alpha}/R$ shows better performance than Min-hop routing in all three scenarios. Note here that $n(VPNs)^{\alpha}/R$ shows insignificant benefits over $1/R$ when β is very small or large. The region of β where $n(VPNs)^{\alpha}/R$ shows significantly better performance is relatively smaller than the case where the Waxman model or real network were used. As before, the performance of $n(VPNs)^{\alpha}/R$ is insensitive to different values of α.



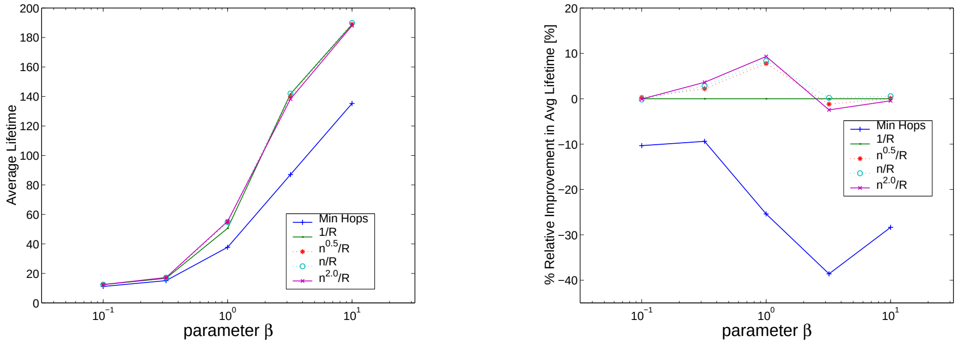
(a) Average lifetime - Scenario 1

(b) Percentage increase in average lifetime - Scenario 1



(c) Average lifetime - Scenario 2

(d) Percentage increase in average lifetime - Scenario 2



(e) Average lifetime - Scenario 3

(f) Percentage increase in average lifetime - Scenario 3

Figure B.3: Results versus parameter β with Pure Random model topology.

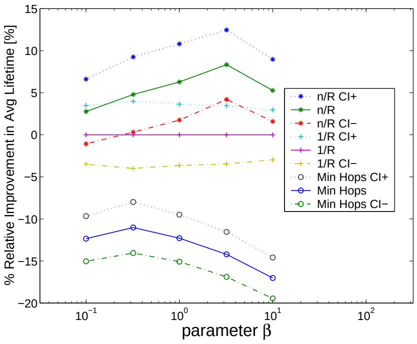


Figure B.4: Confidence interval of percentage increase in average lifetime - Scenario 1 with Pure Random model.

Bibliography

[1] D. Chiu and R. Jain, *Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks*, Computer Networks and ISDN Systems 17, 1989, 1-14.

[2] V. Paxson, *End-to-end Internet Packet Dynamics*, Proc. of ACM SIGCOMM, September 1997.

[3] C. Lefelhocz, B. Lyles, S. Shenker and L. Zhang, *Congestion Control for Best-Effort Service: Why We Need a New Paradigm*, IEEE Network, Juauary/February, 1996.

[4] B. Braden, et al, *Recommendations on Queue Management and Congestion Avoidance in the Internet*, IETF RFC 2309, April, 1998.

[5] B. Davie and Y. Rekhter, *MPLS : technology and applications*, Morgan Kaufmann Publishers, 2000.

[6] T. Ott, T. Lakshman and L. Wong, *SRED: Stabilized RED*, IEEE INFOCOM, 1999.

[7] W. Feng, D. Kandlur, D. Saha and K. Shin, *BLUE: A New Class of Active*

Queue Management Algorithms, University of Michigan CSE-TR-387-99, April 1999.

[8] Stefaan De Cnodder, Kenny Pauwels and Omar Elloumi *A Rate Based RED Mechanism*, The 10th International Workshop on Network and Operating System Support for Digital Audio and Video,(NOSSDAV) 2000.

[9] *ATM Forum Traffic Management Specification version 4.1*, AF-TM-0121.000, March 1999.

[10] S. Floyd and V. Jacobson, *Random Early Detection Gateways for Congestion Avoidance*, IEEE/ACM Transactions on Networking, Vol. 1, No. 4, August, 1993.

[11] N. Li, S. Park and S. Li, *A Selective Attenuation Feedback Mechanism for Rate Oscillation Avoidance*, the Journal of Computer Communications, special issue on QoS-Sensitive Distributed Network Systems and Applications, Volume 24, Issue 1, Jan 2001.

[12] M. Borrego, N. Li, G. Veciana and S. Li, *Congestion Avoidance Using Adaptive Random Marking*, 2000.

[13] R. Jain, S. Kalyanaraman, S. Fahmy, R. Goyal and S. Kim, *Source Behavior for ATM ABR Traffic Management: An Explanation*, IEEE Communication Magazine, November, 1996.

[14] W. Feng, D. Kandlur, D. Saha, K. Shin, *A Self-Configuring RED Gateway*, IEEE INFOCOM, 1999.

[15] N. Li, G. Veciana, S. Park, M. Borrego and S. Li, *Minimizing Queue Variance Using Randomized Deterministic Marking*, submitted to GLOBECOM 2001.

[16] K. Ramakrishnan, S. Floyd, *A Proposal to Add Explicit Congestion Notification (ECN) to IP*, IETF RFC 2481, January, 1999.

[17] *Network Simulator - ns version 2*, http://www.isi.edu/nsnam/ns/.

[18] W. Stevens, *TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms*, IETF RFC2001, January, 1997.

[19] R. Jain, *The Art of Computer Systems Performance Analysis - Techniques for Experimental Design, Measurement, Simulation, and Modeling*, John Wiley & Sons, Inc., 1991.

[20] G. Ash, *Dynamic Routing in Telecommunications Networks*, McGraw-Hill Telecommunications, 1997.

[21] A. Sridharan, S. Bhattacharyya, C. Diot, R. Guerin, J. Jetcheva and N. Taft, *On The Impact of Aggregation on The Performance of Traffic Aware Routing*, ITC'17, Salvador da Bahia, Brazil, September, 2001.

[22] J. Moy, *Ospf version 2. Technical Report*, RFC 2328, IETF, 1998, http://www.ietf.org/rfc/rfc2328.txt.

[23] J. Moy, *OSPF: Anatomy of an Internet Routing Protocol*, Addison-Wesley, 1999.

[24] Y. Wang, Z. Wang and L. Zhang, *Internet Traffic Engineering without Full Mesh Overlaying*, Infocom2001, Anchorage, April, 2001.

[25] B. Fortz and M. Thorup, *Internet Traffic Engineering by Optimizing OSPF Weights*, Infocom2000, Tel-Aviv, March, 2000.

[26] Y. Serbest, S. Park, A. Sang and S. Q. Li, *A Simple Adaptive SVC Caching Scheme for Voice Trunking Over ATM (VTOA) Applications*, Infocom 2000, Tel-Aviv, March, 2000.

[27] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell and J. McManus, *Requirements for Traffic Engineering Over MPLS*, Internet Draft, draft-ietf-mpls-traffic-eng-01.txt, June, 1999, Work in progress.

[28] R. Ahuja, T. Magnanti and J. Orlin, *Network Flows : Theory, algorithms, and applications*, Prentice-Hall, 1993.

[29] C. Villamizar, *Ospf optimized multipath*, Internet draft, IETF, Feb 24, 1999. draft-ietf-ospf-omp-02.txt. Work in progress.

[30] M. Kodialam and T.V. Lakshman, *Minimum Interference Routing with Applications to MPLS Traffic Engineering*, Infocom 2000, Tel-Aviv, March, 2000.

[31] S. Park, Y. Serbest, W. Lau, H. Bi, A. Sang and S. Li, *Multi-Resource Network Planning with Implications to Routing*, 2000.

[32] G. Apostolopoulos, D. Williams, S. Kamat, R. Guerin, A. Orda and T. Przygienda, *QoS Routing Mechanism and OSPF Extensions*, RFC 2676, IETF, 1999, ftp://ftp.isi.edu/in-notes/rfc2676.txt.

[33] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick, *A Framework for QoS-based Routing in the Internet*, RFC 2386, IETF, 1998, ftp://ftp.isi.edu/in-notes/rfc2386.txt.

[34] Z. Wang and J. Crowcroft, *QOS routing for supporting resource reservation*, IEEE Journal on Selected Areas in Communications, 14(7):1228–1234, 1996.

[35] D. Bertsekas and R. G. Gallager, *Data Networks*, Prentice Hall, Englewood Cliffs, NJ, 2nd edition, 1992.

[36] D. Bertsimas and J. N. Tsitsiklis, *Introduction to Linear Optimization*, Athena Scientific, 1997.

[37] The ATM Forum Technical Committee, *Private Network-Network Interface Specification Version 1.0 (PNNI 1.0)*, March 1996. af-pnni-0055.000.

[38] R. Fourer, D. M. Gay and B.W. Kernighan, *AMPL : A Modeling Language For Mathematical Programming*, Boyd and Frase, 1993.

[39] B. Davie and Y. Rekhter, *MPLS:Technology an Applications*, Morgan Kaufmann Publishers, 2000.

[40] E. Rosen and Y. Rekhter, *BGP/MPLS VPNs*, RFC 2547, March 1999.

[41] S. Kent and R. Atkinson, *Security Architecture for the Internet Protocol*, RFC 2401, November 1998.

[42] E. Rosen, A. Viswanathan and R. Callon, *Multiprotocol Label Switching Architecture*, RFC 3031, January 2001.

[43] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, *An Architecture for Differentiated Services*, RFC 2475, December 1998.

[44] S. Rooney, J.E. van der Merwe, S. Crosby and I. Leslie, *The Tempest, a*

Framework for Safe, Resource Assured, Programmable Networks, IEEE Communications Magazine, vol. 36, pp. 42-53, October 1998.

[45] N.G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K.K. Ramakrishnan, and J. Merwe, *A Flexible Model for Resource Management in Virtual Private Networks*, Proceedings of ACM SIGCOMM 1999, September, 1999.

[46] A. Gupta, J. Kleinberg, A. Kumar, R. Rastogi, and B. Yener, *Provisioning a virtual private network: a network design problem for multicommodity flow*, Proceedings of ACM STOC 2001, pp. 389-398, July, 2001.

[47] A. Kumar, R. Rastogi, A. Silberschatz, and B. Yener, *Algorithms for Provisioning Virtual Private Networks in the Hose Model*, Proceedings of ACM SIGCOMM 2001, August, 2001.

[48] D. Karger and M. Minkoff, *Building Steiner trees with incomplete global knowledge*, Proceedings of 41st IEEE FOCS, 2000.

[49] R. Syski, *Introduction to congestion theory in telephone systems*, vol.4 of *Studies in Telecommunication*, Amsterdam: Elsevier Science, 2nd ed., 1986.

[50] G. Guisewite and P. Pardalos, *Minimum Concave-cost Network Flow Problems: Applications, Complexity, and Algorithms*, Annals of Operations Research, 25 (1990), pp. 75-100, 1990.

[51] B. Lamar, *An Improved Branch and Bound Algorithm for Minimum Concave Cost Network Flow Problems*, Journal of Optimizatins 3: pp. 267-287, 1993.

[52] H. Eggleston, *Convexity*, Cambridge tracts in Mathematics and Mathematical Physics No. 47, Cambridge University Press, 1963.

[53] W. Zangwill, *Minimum Concave Cost Flow in Certain Networks*, Management Science, vol. 14, No. 7, March, 1968.

[54] S. Graves and J. Orlin, *A Minimum Concave-Cost Dynamic Network Flow Problem with an Application to Lot-Sizing*, Networks, Vol. 15 (1985), pp. 59-71, 1985.

[55] G. Guisewite and P. Pardalos, *A Polynomial Time Solvable Concave Network Flow Problem*, Networks, Vol. 23 (1993), pp.143-147, 1993.

[56] D. Kim and P. Pardalos, *Dynamic Slope Scaling and Trust Interval Techniques for Solving Concave Piecewise Linear Network Flow Problems*, Networks, Vol. 35 (2000), pp.216-222, 2000.

[57] S. Kim, *An Optimal Establishment of Virtual Path Connections for ATM Networks*, Proceedings of INFOCOM 1995, pp. 72-79, April, 1995.

[58] R. Ahuja, T. Magnanti and J. Orlin, *Network Flows : Theory, algorithms, and applications*, Prentice-Hall, 1993.

[59] M. Kodialam and T.V. Lakshman, *Dynamic routing of locally restorable bandwidth guaranteed tunnels using aggregated link usage information*, Infocom 2001, Anchorage, USA, April, 2001.

[60] Gustavo de Veciana, Sangkyu Park, Aimin Sang and Steven Weber, *Routing and Provisioning VPNs based on Hose Traffic Models and/or Constraints*, 40th Annual Allerton Conference on Communication, Control, and Computing, UIUC at Illinois, 2002.

[61] G. F. Italiano, R. Rastogi, B. Yener, *Restoration Algorithms for Virtual Private Networks in the Hose Model*, IEEE INFOCOM, 2002.

[62] Anupam Gupta, Amit Kumar, Rajeev Rastogi, *Traveling with a Pez Dispenser (Or, Routing Issues in MPLS)*, Proc. 42nd IEEE Symposium on Foundations of Computer Science, 2001.

[63] F.P. Kelly, *Loss Networks*, The annals of Applied Probability, 1:319:378, 1991.

[64] G. Mao, and D. Habibi, *Loss Performance Analysis for Heterogeneous ON-OFF Sources With Application to Connection Admission Control*, IEEE/ACM Transactions on Networking, Vol. 10, No. 1, Feburuary 2002.

[65] F. Kelly, *Notes on effective bandwidths. In Stochastic Networks: Theory and Applications. (Editors: F.P. Kelly, S. Zachary and I.B. Ziedins)*, Royal Statistical Society Lecture Notes Series, 4, pages 141-168, Oxford University Press, 1996.

[66] S. A. Plotkin, *Competitive Routing of Virtual Circuits in ATM Networks*, IEEE Journal of Selected Areas in Communications, Vol 13, No 6, pp.1128-1136, 1995.

[67] R.L. Easton, P.T. Hutchinson, R.W. Moncello, R.W. Muise, *TASI-E communications system*, IEEE Trans. Commun., pp. 803-807, 1982.

[68] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, *How to Model an Internetwork*, IEEE Infocom, vol.2, pp. 594-602, 1996.

[69] X. Su, and G. de Veciana, *Dynamic multi-path routing: Asymptotic approximation and simulations*, ACM Sigmetrics 2001, 25-36, 2001.

Vita

Sangkyu Park was born in Pusan, Korea on January 7, 1967. He did his B.S. and M.S. at Seoul National University in Electrical Engineering (Control and Instrumentation division) in 1989 and 1991 respectively. He worked as a research engineer from 1991 to 1996 at the Central Research Institute of Hyundai Heavy Industries Co. He joined the University of Texas at Austin as a Ph.D. student in 1996. In 2000 from January to July he worked at SBC Technology Resource Inc. as full-time internship.

Permanent Address: 3105 South IH-35 #3072
Austin, TX 78741

This dissertation was typeset with LaTeX 2_ε[1] by the author.

---

[1] LaTeX 2_ε is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this dissertation were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin, and extended by Bert Kay and James A. Bednar.