The Dissertation Committee for Xun Su

certifies that this is the approved version of the following dissertation:

Flow-based Dynamic Routing In Uncertain Network Environments

Committee:

Gustavo A. de Veciana, Supervisor

Aristotle Arapostathis

Ross Baldick

Edward J. Powers

David P. Morton

Flow-based Dynamic Routing In Uncertain Network Environments

by

Xun Su, MSEE

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

August 2002

To my parents, with love

Acknowledgments

This dissertation summarizes my work in the last few years at UT Austin. Without the help from many people this work would not have been possible. First of all, I would like to express my deepest gratitude to my adviser, Dr. Gustavo de Veciana, who has guided my way over the years. He has shown me by example what it means to be a scholar, a teacher and a professional. I would also like to sincerely thank Dr. Aristotle Arapostathis, Dr. Ross Baldick, Dr. Edward Powers and Dr. David Morton for serving on my committee and providing valuable insights for my research.

I am fortunate to have so many wonderful colleagues and friends. The discussions with Jay Yang, Steven Weber, Xiangying Yang, Jangwan Lee, Philip Girolami, Tae-jin Lee, Mike Montgomery, Aimin Sang, Sangkyu Park have been delightful and I learned a lot from them. I am indebted to Dr. Ching-Fong Su, who was first my colleague at UT and later became my mentor at Fujitsu Labs during my summer internship. I thank him for his keen ideas and generous help.

But mostly, I owe all this to my parents. They have always been there for me, through it all, no matter what. Their unconditional love and selfless support are what make me here today. This dissertation is dedicated to them.

XUN SU

The University of Texas at Austin

August 2002

Flow-based Dynamic Routing In Uncertain Network Environments

Publication No. _____

Xun Su, Ph.D.
The University of Texas at Austin, 2002

Supervisor: Gustavo A. de Veciana

In this dissertation we study flow-based dynamic routing schemes. We propose routing schemes to achieve good network level performance such as flow blocking rate, and user level performance such as flow throughput. The design principle is to route a flow so that we can satisfy its Quality of Service (QoS) requirement, and minimize the negative impact of the routing on the performance of current and future flows of the network. To this end we identify a number of operating regimes in which user demands, network states, and routing decisions interact. We propose different routing mechanisms based on the unique properties of the operating regimes.

For the networks operating in a regime in which link states are highly dynamic and traffic demands are bursty, we propose a dynamic multi-path routing scheme to disperse traffic between a source and a destination. We investigate the critical issue of how to select the set of least congested paths over which to disperse traffic, and examine the performance of this routing scheme in a mesh network. The performance of our dynamic multi-path routing scheme is quite robust with respect to various operating parameters and it offers consistent performance improvement over the baseline single path routing scheme.

We then study dynamic routing schemes in the networks where network states evolve on a modest timescale which allows a user to predict its performance during its sojourn in the network. We model the link load dynamics and propose a routing scheme that jointly consider the current link load, the flow holding time, the average link load and mean reversion for the load dynamics. This routing scheme is beneficial for a range of applications where efficient fair-sharing of the excess network resources can improve network and user performance. We show our routing scheme, designed to improve users' average performance during their sojourn in the network, indeed leads to an efficient fair use of network resources.

We also investigate routing algorithms for providing failure protection. The objective is to efficiently provision network resources so that when certain failures occur we can re-route the affected traffic with minimal service disruption. We observe that while a protection path for a given failure is constrained since it has to select its detours around the failure, it can nonetheless share the protection resources with protection paths for other failures. We identify a routing metric which effectively captures the sharing potential in the network. This routing metric and the associated routing algorithms, are shown to provide failure protection with significantly reduced resource redundancy.

Contents

Bibliography 100

List of Tables

List of Figures

Chapter 1

Introduction

In order to provide end-to-end quality of service (QoS) guarantees to users, future networks are likely to require enhanced traffic control mechanisms. Indeed, the current best effort service model employed in the Internet, though simple and robust, is not particularly well-suited to address *end-to-end* QoS requirements. Instead, we believe that in order to satisfactorily meet QoS demands it is essential to deploy a well-planned network control infrastructure including routing, congestion control and packet scheduling mechanisms. One might argue that the best effort service will suffice if a large amount of bandwidth is always available, *i.e., over-provisioned.* However, we believe that the anticipated onslaught of the traffic demands, possibly brought on by the popularization of broad-band access services, *e.g.,* ADSL and cable modem, will soon fill up any bandwidth void. Moreover, even if bandwidth is abundant, there might still exist a need to differentiate the level of service provided to various users. This in turn suggests that the implementation of a system of auxiliary traffic control mechanisms, such as traffic classification and advanced routing schemes, will become increasingly important. A prime example of the ongoing efforts in this direction is MPLS[6], whose emphasis on Traffic Engineering has contributed to vigorous research activity in the QoS domain, *e.g.,* Constraint-based Routing[9].

In this dissertation we focus on the design and performance evaluation of dynamic routing strategies. Specifically, we note that the *hop-by-hop* routing mechanism, now ubiq-

uitous in the Internet, is not an ideal choice for developing QoS-aware routing algorithms if the QoS guarantee is to be ensured on an end-to-end basis. By contrast, *source routing* might be better suited to select paths satisfying such QoS requests, but it requires the availability and distribution of a large amount of network state information, possibly resulting in scalability problems. Moreover, since the state of the network is in constant flux, routers may have to make routing decisions based on uncertain state information. In particular, future routing mechanisms may use hierarchical aggregation of state and/or topology information to deal with scalability issues, naturally resulting in a loss of accuracy. By the same token, routers may also have incomplete information concerning the characteristics of the users' traffic. Indeed, since traffic is often best modeled as a stochastic process, there is a high degree of uncertainty in specifying traffic via crude parametric source models and then translating these to an end-to-end QoS guarantee. In light of these observations, our goal is to investigate source routing schemes based on uncertain network and source information, that are geared toward providing QoS guarantees, or improving user experience, by optimizing the resource usage among contending flows. In the following we identify a number of network operating regimes that may arise in practice and propose routing solutions that cater to the distinct characteristics of these regimes.

Dispersing Highly Dynamic Traffic. Let us first consider the case where the network operates in a regime where link loads are highly dynamic and hence the link load dynamics are essentially unpredictable. In particular, this might be caused by a large number of bursty short flows "flying" through the network, *i.e.,* they arrive and depart on a short timescale. In this case the actual (versus available) link states encountered by consecutive flow arrivals/routing requests might be quite different. Moreover, since these flows are processed in a *distributed* manner, in the sense that they are handled by different routers without a central controller to coordinate decision processes, the advertised link states will tend to "synchronize" the routing of the traffic flows, *i.e.,* they will collectively arrive and then depart a given set of links depending

on the specific load configuration in the network. This further increases the link load fluctuation which makes maintaining up to date link states difficult. A natural strategy to combat this pathological condition is to use randomization to "de-synchronize" the routing decisions at various routers in the network. Specifically, we propose to take advantage of the inherent spatial-temporal diversity in the underlying traffic and network, and disperse the traffic flows between a source node and a destination node through a set of properly selected routes. This would allow us to maximize the utilization of network resources while minimizing the negative impact that traffic dispersion might have on the performance of the traffic flows concurrently traversing the network, and those that will arrive in the future. As discussed in the sequel, this approach to routing leads to a form of *dynamic multi-path* routing. In the scenario we envisage the flows arrive and depart at a fast pace, so a sensible design objective is to allow the maximum amount of traffic to get through, rather than trying to optimize the performance perceived by the individual flows. As we will see, routing decisions aimed at optimizing the above objective may or may not coincide with conventional shortest path routing decisions. To effectively design such a dispersion-based routing scheme we will address the critical issue of how to select the *set* of paths over which to perform traffic dispersion/routing.

Predictive routing to improve flow-perceived performance A unique opportunity exists to improve the user-perceived QoS in the network operating regime where link state and traffic flows both evolve at modest timescales. Unlike the previous case where the traffic flows are highly dynamic and the link load dynamics are unpredictable, in this case a certain level of knowledge on the flow characteristics and network dynamics can be used to guide routing mechanisms. It is possible in this regime to improve user-perceived performance by estimating what flows will experience during their sojourn in the network. Let us consider the following generic service model, as it encompasses a number of traffic classes in reality, *e.g.,* ATM VBR[19] and

rate adaptive applications[58]: the incoming flow asks for a minimum bandwidth requirement from the network, if the network can satisfy the request, the flow is admitted into the network and goes on a route that is perceived of good quality; if there are no resources to satisfy the flow's minimum bandwidth request on the route selected based on the available information, the flow is rejected. Furthermore, after admission into the network, if there are additional resources available, the flow will share resources with other ongoing flows. Hence there are two objectives, (1) to satisfy users' minimum QoS requirements and (2) if possible, to provide them with additional QoS benefit from sharing available bandwidth. We believe that in most cases the second objective complements the first objective. Thus a realistic goal would be to admit as many flows as possible with their minimum QoS requirement and then allow them to share the extra bandwidth to the best possible effect. In short, a good routing scheme should distinguish itself by realizing good user-perceived QoS without having to seriously compromise flow blocking rate.

The performance of the network operating in this regime will depend on the changing link states and the manner in which routing mechanisms respond to these changes. In this regime, link loads fluctuate on "modest" timescales, *i.e.*, flows stay in the network on timescales comparable to the link load dynamics. This suggests that during their sojourn in the network, flows would benefit from extra amount of bandwidth shared with other flows. Thus in this regime a good routing mechanism should not only allow the network to carry a large amount of traffic volume, but also improve the perceived performance for admitted flows. To achieve performance goals in these respects we need to effectively model the link load dynamics, and efficiently construct routing metrics that can take advantage of the additional information. More specifically, we believe that in this regime one might expect link dynamics to evolve in a *predictable* fashion, that can be modeled to enhance routing mechanisms. This allows us to design routing schemes in a more informed manner, using the link state information

such as the mean load and the rate at which link loads reverse to their mean, in addition to the traditional notion of "current load". This information can be extracted by a parametric estimation technique and incorporated into the routing algorithms by accounting for traffic characteristics, *e.g.*, expected flow holding times, and their willingness to adjust to the available network resource, *i.e.*, elasticity.

Provisioning protection wavelength for failure restoration A further challenge arises when we consider routing mechanisms for networks in which emergencies such as fiber cut or switch malfunctions might occur. In this case the network should be able to respond to the emergency in a prompt manner, with minimal degree of service disruption inflicted on ongoing traffic. Usually, one can either employ a *protection* scheme or a *restoration* mechanism. The difference lies in the timing of the recovery actions, *i.e.*, pre-provisioning or real-time response. In a *protection* scheme, the network sets aside a certain amount of resource (*e.g.*, wavelength, or MPLS label) for the purpose of protecting ongoing traffic flows and upon failure events the disrupted traffic is switched over to the protection resource. In a *restoration* scheme, no extra resource is set aside for recovery purposes, and if a failure occurs the network has to seek out an alternate path on the fly. In general, the *restoration* scheme is more resource-efficient but takes longer to recover from a failure, if there are enough resources in the network at the moment of failure. By contrast, the *protection* scheme is more responsive and guarantees failure recovery, but often at the cost of additional resource consumption. We believe the occurrence of network failures are essentially unpredictable, thus the routing/protection design should aim at long-term event horizons where failures might eventually occur. The emphasis is therefore placed on provisioning *sufficient* protection resources so that *all* possible failures are safely protected. Evidently, in this design the protection takes higher priority, but it need not exclude consideration of resource efficiency. Since backup resources are unused when the network operates without failures, they should be allocated in a cost-effective manner if it is possible.

In reality, the protection/restoration problem is both important and difficult since one has to take into account a number of complex factors when devising an operational strategy, *e.g.*, number of optical ADM versus SONET ADM, traffic grooming, wavelength conversion capabilities (or the lack thereof), multi-layer protection/restoration coordinations. We take a first step in this problem domain and propose an efficient mechanism to provision the restoration bandwidth. The idea stems from the realization that the network failures are inherently rare events, thus they are very unlikely to occur concurrently. Hence the protection resources can be shared among potential failure events. Our main contribution lies in identifying a novel link metric, which along with an associated routing algorithm, realizes the bandwidth sharing potential in the WDM networks in a dynamic online routing context.

In the sequel we propose a number of solutions/algorithms to address the above challenges. These algorithms share some common features, *i.e.*, they are all *distributed*, *approximate* and *online*. This contrasts with the *centralized* algorithms commonly used in other optimization problems, *e.g.*, network design and bandwidth provisioning. This is in large part due to the *dynamic* nature of the problem we tackle, *i.e.*, traffic flows arrive and depart, and only aggregated link states are available, *i.e.*, overall link loads rather than detailed information on per-flow route selection. In this case formulating and solving a quasi-static optimization problems with detailed per-flow routing information become both infeasible and irrelevant. Moreover, we believe it is unrealistic to assume a consistent knowledge of the entire traffic matrix by the individual routers (or even by a central controller), not to mention the exact sequence of the arrivals. Under such circumstances, it makes sense to design dynamic distributed online routing schemes that closely approximate the optimal performance value. Furthermore, we observe that traffic demands are intrinsically heterogeneous, *i.e.*, they have different source, destination, holding time, QoS requirement, etc.. The effective approaches to alleviating this additional level of complexity include developing multi-service extensions to the loss network framework used to analyze telephone

networks [25, 49, 62]. Nevertheless, new problems arise when one considers routing in networks where guaranteed and best effort services share resources [42]. Specifically, it will be of interest to investigate how the heterogeneity in the traffic demand impacts overall routing efficiency, *e.g.*, flow blocking rate, and/or the performance experienced by different classes of users, *e.g.*, user's achieved throughput.

The dissertation is organized as follows. In Chapter 2 we present a study of dynamic multi-path routing scheme addressing a network regime where traffic arrivals and network states are highly dynamic. In Chapter 3 we study performance of a single path routing scheme utilizing expected flow holding time, and the estimates of the link load dynamics where link load dynamics operate on modest timescales enabling one to optimize the user-perceived QoS. In Chapter 4 we examine issues involved in provisioning protection bandwidth in the WDM networks and propose a solution to the problem. We conclude the dissertation in Chapter 5 discussing future research directions.

Chapter 2

Dynamic Multi-path Routing: Asymptotic Approximation and Simulations

2.1 Introduction

As the Internet continues growing and new technologies emerge to meet this growth, networking researchers are faced with the increasingly daunting task of controlling and/or managing extraordinary amounts of traffic. Traditionally, network operators have relied on buffers at network nodes and/or congestion control to deal with fluctuations in traffic loads. However, as traffic loads become more bursty, the size and the speed of the buffers in the network are not growing commensurately with the link speed, making buffering techniques less effective. Moreover, although link speeds increase dramatically, propagation delays stay roughly unaffected, calling into question the effectiveness of flow/congestion control mechanisms. Indeed, to control the congestion inside the network, we have traditionally relied on end-to-end reactive flow control schemes, *e.g.*, TCP[33] and RED[18]. These schemes rely on coordination among links within the network and traffic sources at the network edge. Links detect the congestion and send back "congestion indications" (*e.g.*, drop/mark packets) to the traffic sources which in turn respond by adjusting their transmissions. However, in an operating regime with high bandwidth-delay product, *i.e.*, transmission rate times the round trip time from the ingress node to the egress node, this

reactive approach is not effective. The problem is twofold: (1) since the link capacity is huge, the traffic in flight when a congestion signal is generated is enormous so the network must be able to buffer a large amount of data and (2) since access speeds may be very high, a traffic burst that induces congestion may finish by the time the traffic source receives the corresponding congestion indications. In both cases the response occurs too late to effectively avoid congestion. A similar phenomenon is observed in the dynamic routing context, exemplified by the routing "synchronization" problem, where link updates are late and ineffective in navigating the traffic flows across a congested network. An interesting question that stands out, is whether we can avoid network congestion without having to slow down the user's transmission rates.

In this chapter we focus on an operating regime in which traffic flows come and go frequently within the timescale of link state advertisements. We view such flows as high speed transmissions, *i.e.,* a sequence of IP packets transmitted at a high rate, and following the same path. As a result, network congestion in this context exhibits a relatively short term dynamics and can not be effectively controlled through per source feedback schemes like TCP. Instead of slowing down user transmission rates to enable better congestion control, we propose routing schemes that alleviate network congestion while allowing users to send traffic at their full access rates. The idea is to disperse traffic flows sharing the same ingress/egress points via multiple paths on the network, in order to achieve statistical multiplexing of the flows over available network resources [28]. This in itself is not a new idea, and is part of a tradition of alternative routing and dispersion used in some circuit switched networks [40, 52, 59, 55, 35, 22, 63].

Traffic dispersion, with its early origins in "dispersity routing"[46], has been an active research area. Dispersion at the packet, burst and flow/connection levels have been considered, see [28] and the references therein. In particular, [46] originated the idea of packet-level dispersion in the context of store-and-forward data networks, and showed that by spreading the traffic over two (or three) paths the average delay of a message is sig-

nificantly reduced. Dispersity routing at flow/connection level was further adapted to the ATM networks[47], where it has been shown to equalize traffic loads and increase overall network utilization for short flows with durations in the same order as the propagation time or less. This work also points out the possible benefits of dispersing flows adaptively. The combination of these two issues, short flows and adaptive multi-path routing, is the starting point for our study. However, as pointed out in [28], the problem of determining the optimal set of paths over which such dispersion should be performed, remains open.

Somewhat akin to this problem, alternative routing and trunk reservation have been studied extensively, see *e.g.,* [40, 52, 59]. In the context of circuit-switched networks [22, 55, 35], a trade-off is sought between increasing routing options and resource utilization. This means that if the primary (usually short) paths experience congestion, secondary paths will be used to carry the traffic load. However, secondary paths are only used if they are not loaded beyond a certain *threshold*, otherwise new arrivals are blocked. The key parameters, the primary/secondary paths and the *threshold*, are difficult to optimize for general network topologies.

This chapter examines one of the key issues that needs to be addressed in dispersing flows over multiple paths, *i.e.,* how to (dynamically) select the set of candidate routes over which traffic flows will be dispersed based on potentially outdated link state information, or even adapt this set to achieve better overall performance. This contributes to the ongoing research efforts that extend the functionality of the OSPF[65], MPLS[64], or Diffserv[60], where deterministic[65, 64] or probabilistic[60] header processing (*i.e., hashing*) mechanisms have been proposed to facilitate the dispersion of traffic flows over multiple paths from an ingress node to an egress node. Our study provides sensible routing decisions based on which packet-level hashing decisions can be constructed, *i.e.,* the set of paths over which the packet flows are sent. By appropriate hashing operations packets with the same attributes (*e.g.,* source address, destination address, QoS requirement) will form a flow that traverses the *same* path.

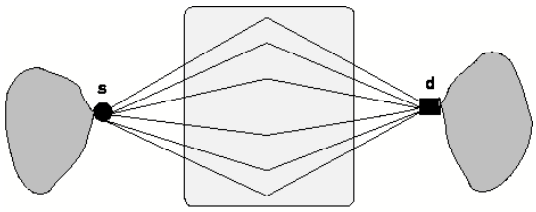

Figure 2.1: An abstract model of route selection.

In the ensuing sections we first consider a simple model consisting of parallel links between an ingress-egress node pair. The main result suggests a simple and robust policy to select a subset of candidate links over which to spread incoming traffic flows. We then propose and evaluate a dynamic multi-path routing scheme for mesh networks.

2.2 A stochastic parallel-link model

2.2.1 Problem setup

In this section we study the idealized model shown in Figure 2.1, where a pair of ingress and egress nodes are interconnected via a set of n links, $L = \{1, 2, \cdots, n\}$, each having the same capacity $c_l = c$, for $l \in L$. Notice that these parallel links can be used to model either real links, or *disjoint* routes between an ingress node and an egress node. Let λ_{sd} denote the flow arrival rate from node s to node d. Without loss of generality we assume each flow transmits packets at a fixed unit rate for a random duration with mean μ^{-1}, along the route it is assigned. The offered load associated with node s and node d, is thus λ_{sd}/μ. The traffic load on link $l \in L$ at time t, denoted by $x_l(t)$, is the sum of the total number of flows currently routed across it. We let $\hat{x}(t) = (x_1(t), x_2(t), \cdots, x_n(t))$. The flow arrival rate to link l is denoted by γ_l, *i.e.*, the part of total arrivals between node s and node d, *i.e.*, λ_{sd}, that is routed to link l. As an approximation, in the sequel we examine the dynamics of the link loads via a fluid model. The flow departure rate is proportional to the current link

load $x_l(t)$ and is given by $x_l(t) \cdot \mu$. We consider a time scale of interest, t, that represents the potential delays involved in updating link states. To model the fact that the routing decisions are made based on outdated information, we assume traffic arrivals during $[0, t]$ are routed based on the link state $\hat{x}(0)$ available at time 0. Under a routing scheme which results in a total traffic arrival rate γ_l to link l and does not lead to link overflows, the link state $x_l(t)$ tracks the following differential equation,

$$\dot{x_l}(t) = \gamma_l - x_l(t) \cdot \mu, \tag{2.1}$$

hence

$$x_l(t) = x_l(0)e^{-\mu t} + \frac{\gamma_l}{\mu}(1 - e^{-\mu t}), \tag{2.2}$$

where $x_l(0)$ is the state of link l at the beginning of the time interval $[0, t]$, and γ, *i.e.,* the routing/assignment of incoming traffic flows to link l, remains fixed over $[0, t]$.

We use an additive network congestion measure, $s(x(t)) = \sum_{l=1}^{n} f(x_l(t))$, where $f(x_l(t))$ is an increasing and convex function of $x_l(t)$, *e.g.,* $f(x_l(t)) = \frac{1}{c - x_l(t)}$. Our goal is to find an allocation of incoming traffic flows within a time interval $[0, t]$ to the n links such that the increase of the system congestion measure, *i.e.,* $s(x(t)) - s(x(0))$, is minimized. This is equivalent to minimizing the system congestion $s(x(t))$ *at time t.* This choice is intended to simplify the analysis, as will be seen in the following section. Eq.(2.1) captures link load dynamics that may impact the routing decisions, *i.e.,* the more link l is loaded, the faster traffic flows depart from it. Intuitively, this observation suggests $x_l(t)$ may underestimate the available capacity on a heavily loaded link when it comes to routing new traffic demands. In particular, the capability of a more loaded link to accommodate traffic demands might be favorably "upgraded" since it is likely to see more departing flows.

Given the current link states $\hat{x}(0)$ and the offered load λ_{sd}, one can in principle determine the optimal routing that minimizes system congestion. Our goal, however, is to find a simple dynamic multi-path routing policy. In particular, we focus on a form of least-loaded routing scheme where *equal shares* of the traffic load are routed on k out of the n

links[1]. The key problem is to determine an "optimal" k which is "robust" with respect to a range of possible link loads, the intensity of the flow arrival process, and the mean flow holding time. In the sequel, we derive such a solution by finding a k which on average is "optimal" over a range of possible link states. We call this *dynamic multi-path routing* since based on the network state k links are selected to disperse the traffic.

Analysis

We let X_l, $l = 1, 2, \cdots, n$, denote the random loads on the links *at time 0*. We assume that they are independent and identically distributed with a continuous distribution function F and support set $[0, c]$. These distributions might be selected to reflect the typical operating regimes of the system, e.g., typically lightly loaded or heavily loaded. Alternatively one might select the prior distributions on the link loads to be uniformly distributed, so as to achieve a robust solution over a range of possible operating regimes. Note that in reality a number of factors impact F, *e.g.,* traffic arrival rates, flow holding times, and routing algorithm, thus our assumption on the random link loads may not be realistic in practice. In particular, the link loads are neither identically distributed, nor independent. However, our intent in introducing this simplifying assumption, is to enable analytical derivation of a robust policy and in turn garner interesting insights on dynamic multi-path routing.

Since our policy involves selecting the k least-loaded links, we will make use of *order statistics* on link loads. We use $X^n_{(i)}$ to denote the i^{th} order statistic of the n link loads at time 0, thus $X^n_{(1)} \leq X^n_{(2)} \leq \cdots \leq X^n_{(n)}$. We let $\lambda_{sd} = \lambda \cdot n$, so λ is a measure of the flow arrival rate, normalized by the number of *options, i.e.,* n, over which routing decisions are to be made. Suppose incoming traffic flows over a time interval $[0, t]$ are spread over the k

[1]We opt to focus on this scheme, due to the simple cyclical implementation it implies in a highly dynamic environment, as opposed to, *e.g.,* routing weighted shares of traffic to different links, in which case a set of weights have to be *dynamically* maintained.

links that are the least-loaded at time 0 , then the resulting congestion increase is

$$D^n(k) := \sum_{i=1}^{k}[f(X^n_{(i)}e^{-\mu t} + \frac{n\cdot\lambda}{k}(1-e^{-\mu t})) - f(X^n_{(i)})] + \sum_{i=k+1}^{n}[f(X^n_{(i)}e^{-\mu t}) - f(X^n_{(i)})].$$

The first sum on the right hand side accounts for the change in the congestion level for the k least-loaded links which share the incoming traffic flows during time interval $[0,t]$, while the second term corresponds to the links which see no additional load.

An "optimal" selection of k might correspond to solving the following minimization problem:

$$k^* = \text{argmin}_k\{E[D^n(k)] \mid k \in \{1,2,\ldots,n\}\}. \tag{2.3}$$

Note that for a given set of link loads it is possible that $k \neq n$, *i.e.,* dispersing traffic equally everywhere might not be optimal. Moreover, problem (2.3) accounts for the dynamics of the system, *i.e.,* flow arrivals *and* departures. As a result a more accurate estimates of the links' traffic-accommodating capability are used. Finally, we take the expectation so as to obtain a choice of k that is optimal "on average" over a range of possible link loads.

However, as it stands problem (2.3) is still quite difficult to solve. In the following we consider an asymptotic regime, wherein the offered load $n\cdot\lambda$ and number of links n grow [2], *i.e.,* we consider a sequence of networks with increasing routing diversity and carried load. We parameterize k as $k = \lceil \alpha n \rceil$, hence α corresponds to the fraction of (least-loaded) links over which the traffic flows will be spread. Our goal is to find α which minimizes the *normalized* average congestion increase as $n \to \infty$, *i.e.,*

$$\min_{0\leq\alpha\leq 1} \lim_{n\to\infty} \frac{E[D^n(\lceil \alpha n \rceil)]}{n}. \tag{2.4}$$

The following theorem establishes that (2.4) can be expressed in two equivalent forms that are amenable to analysis. The proof is deferred to the appendix.

[2] This scaling might correspond to the practical context where additional *wavelengths* are added to an optical fiber to increase its total capacity. Each of these added wavelengths can be thought of as an additional link in our model.

Theorem 2.2.1 *The problem defined in (2.4) can be rewritten as*

$$\min_{0\leq\alpha\leq 1}\left\{E[f(X\cdot e^{-\mu t}+\frac{\lambda(1-e^{-\mu t})}{\mu\alpha});X\leq F^{-1}(\alpha)]+\ E[f(X\cdot e^{-\mu t});X\geq F^{-1}(\alpha)]\right\}, \quad (2.5)$$

or equivalently, as

$$\min_{0\leq y\leq c}\left\{E[f(X\cdot e^{-\mu t}+\frac{\lambda(1-e^{-\mu t})}{\mu F(y)});X\leq y]+\ E[f(X\cdot e^{-\mu t});X\geq y]\right\}, \quad (2.6)$$

where the optimal decision variables α^* and y^* are related by $\alpha^* = F(y^*)$. Here F is a continuous distribution on $[0, c]$ modeling the link loads on the parallel-link network.

■

Observe that the theorem suggests that it is *asymptotically* equivalent to use the $\alpha^* \cdot n$ least-loaded links or all the links with load less than y^*, where α^* and y^* are the optimizers of problems (2.5) and (2.6), respectively. This follows by a simple change of variables. However, note that in practice these correspond to two different *modes* of routing, *i.e.*, routing on $\alpha^* \cdot n$ least-loaded links vs. routing on a set of links whose loads are below a threshold y^*.

Assuming the prior distributions of link loads are independent and uniform, the optimal choices for α^* and y^* can be determined. The proof of the following fact can be found in the appendix.

Fact 2.2.1 *Suppose the link loads are uniformly distributed on $[0, c]$, then the minimizers for (2.5) and (2.6) are respectively*

$$\alpha^* = \min\{\sqrt{\frac{\lambda}{\mu c}\frac{(1-e^{-\mu t})}{e^{-\mu t}}}, 1\},$$

and

$$y^* = \min\{\sqrt{\frac{\lambda c}{\mu}\frac{(1-e^{-\mu t})}{e^{-\mu t}}}, c\}.$$

■

Note that for uniformly distributed link loads, the optimal parameters α^* and y^* are not sensitive to the exact form of the system congestion measure f, it need only be increasing and convex.

Based on Fact 2.2.1 we can make a number of interesting observations.

Observations

As the traffic arrival rate λ increases, or the mean flow holding time μ^{-1} decreases, or λ increases for a fixed offered load $\rho = \lambda/\mu$, one disperses the traffic flows over a *larger* set of paths. Hence as an engineering guideline, it makes sense to differentiate the operational parameters at various network nodes with different types of incoming traffic requests. In particular, a network node should actively route its connection requests over *a large set of paths* if these requests are mostly *short and arrive frequently*, or direct its connection requests to the *least-loaded* path if these requests are mostly *long and arrive infrequently*.

The intuition for the above observation, lies in that the system congestion measure $s(x(t))$ is a symmetric sum of increasing convex functions. This suggests that to approach optimality one should route the traffic flows so that link loads at time t are balanced. From (2.2) we see that when if the flow departure rate is large the differences in the *initial* link loads should be "discounted" and we should "perceive" all link loads as approximately "equal." In this context the natural routing decision to make, in order to bring the link loads at time t close to each other, is to spread the incoming traffic flows over a large set of links.

Suppose one scales the offered load ρ (while keeping mean flow holding time μ^{-1} fixed) and link capacity c in proportion, then the optimal fraction of links over which one should route the traffic remains unchanged. This suggests that in practice if we build up the network capacity and the traffic load grows in proportion then α^* and y^*, *i.e.,* the range of paths over which we route traffic flows, remain fixed. On the other hand, we should adjust the range of multi-path routing mechanism if the rate of capacity expansion does not match that of the traffic growth. More specifically, one might need to modify the operational

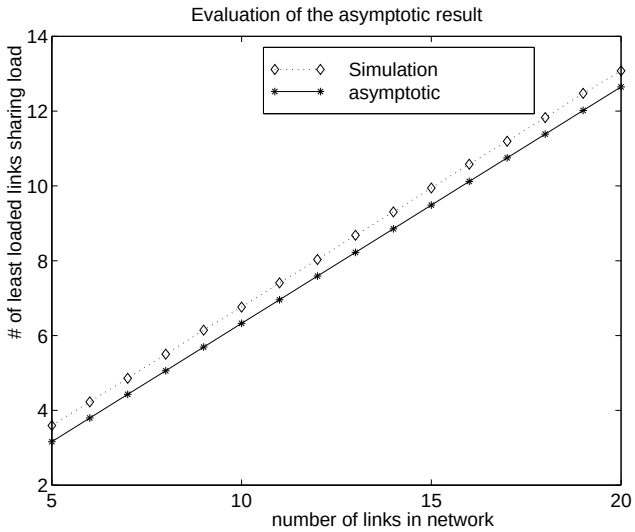

Figure 2.2: The least-loaded links used to spread traffic: increasing total links.

parameters α^* and y^* if user traffic outgrows the network capacity.

Now suppose the time scale of interest t grows, *e.g.*, one has to limit link state advertisements. Notice that as t increases the impact of the initial link load diminishes. Intuitively, if this is the case, the optimal allocation is to spread the load evenly among a large set of links. This is verified by the result in Fact 2.2.1.

In practice, we might not only have incomplete knowledge of the link states but also of the arrival rate λ. From Fact 2.2.1 we see that the optimal parameters α^* and y^* are square root functions of λ. This suggests that these optimal parameters are relatively insensitive to the exact value of λ as the routing diversity increases, thus a reasonable estimate may suffice.

Validation

In the previous section we obtained an asymptotic result concerning the number of links over which to disperse traffic, in a regime where the load and number of links (disjoint paths) grow in proportion. Here we evaluate the quality of the result, in the case where

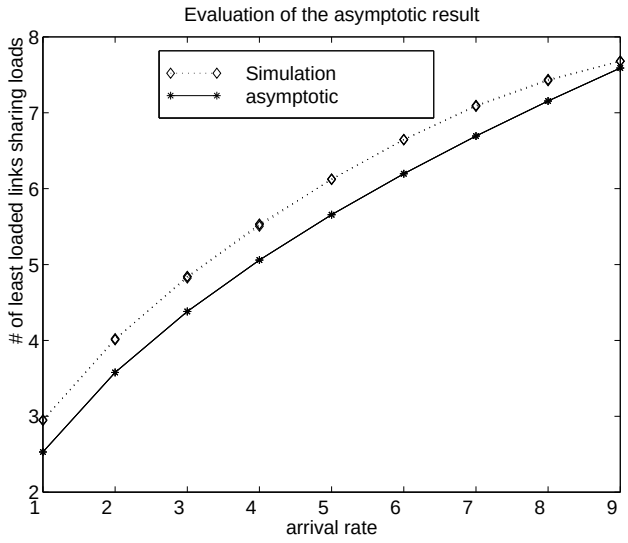

Figure 2.3: The least-loaded links used to spread traffic: increasing arrival rates.

n is small, or modest, via simulation. Suppose $X \sim$ Uniform$[0, 10]$, $\lambda = 4$, $t = 1$, and $\mu = 0.1$. We compare the number of least-loaded links that would be selected based on the asymptotic model versus the corresponding values obtained by successively sampling the link loads and computing the averaged optimal number of links over which to disperse traffic. In Figure 2.2 we exhibit the comparison between theoretical/asymptotic and simulated average optimal choices, in terms of the optimal number of least-loaded links over which to route the traffic flows. We observe that the number of least-loaded links obtained via the asymptotic formula is within 10% of the average obtained via simulation, where the number of links n ranges from 5 to 20. In Figure 2.3 we evaluate the effectiveness of Fact 2.2.1. Here we fix the number of links $n = 8$, $t = 1$, $\mu = 0.1$, $X_i \sim$ Uniform$[0, 10]$, and increase λ from 1 to 9. The asymptotic prediction matches its simulated counterpart to a satisfactory degree. We conclude that the asymptotic result provides a reasonable approximation to select the k links over which to disperse traffic flows.

Realizing dynamic multi-path routing — three alternatives

In the previous sections we explored a dynamic multi-path routing scheme, where equal shares of the traffic load were routed over a dynamically selected set of network links. An asymptotic analysis on how to dynamically select such links is captured by (2.5) and (2.6). We can interpret the solutions to these optimization problems as suggesting two different schemes: routing over the $k = \alpha^* n$ least-loaded paths (or shortest paths, if we equate the *length* of a link to its load), or routing over all the paths that have a load less than y^*.

The implementation of the first scheme corresponds to the classic k-shortest path algorithm[73]. We shall call the first scheme DKSP, which is short for Dynamic k Shortest Paths. Note that this scheme spreads the traffic flows over k links/paths with potentially different loads. This is in contrast with the traditional least-loaded routing scheme, where one randomly selects a link only among those with least load.

For the second scheme, notice that for a given parameter y^* and a particular set of network loads there may not be any candidate links with load less than y^*. To address this problem, we consider two solutions that correspond to 1) a *pre-determined* link load quantization mechanism, named DQSP, which is short for Dynamic Quantized Shortest Paths, and 2) a *dynamic* threshold mechanism, named DTMP, which is short for Dynamic Threshold Multi-Paths.

DQSP can be interpreted as a link state quantization scheme. In particular, the ingress nodes (or the links) quantize the link loads based on a threshold y^*, and traffic is routed over the links with the least quantized load. Figure 2.4 illustrates two scenarios where each "$\times$" indicates the amount of load on a given link, and the circled links are those over which traffic flows will be spread based on the threshold y^*. It is evident that by quantizing the link load, we can increase the number of links that are "equally" loaded. Notice that under this scheme if there is no link with load less than y^*, we can use all the links to route the traffic flows. This can be refined as follows. Conditional upon all the link loads exceeding y^*, we can formulate a modified version of the previous optimization problem,

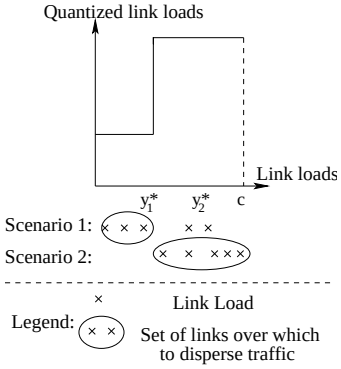

Figure 2.4: A two-level quantizer.

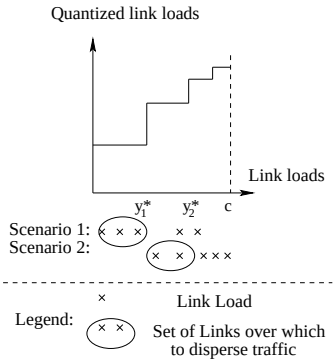

Figure 2.5: A multi-level quantizer.

and recursively obtain a sequence of quantization thresholds: $\vec{y}^* = (y_1^*, y_2^*, \cdots, y_n^*)$, where

$$y_1^* = y^*,$$
$$y_{i+1}^* = y_i^* + \sqrt{\frac{\lambda \cdot (c - y_i^*) \cdot (1 - e^{-\mu t})}{\mu \cdot e^{-\mu t}}}, \quad i = 1, 2, \cdots, n.$$

The procedure terminates at the index i where $y_i \leq c$ and $y_{i+1} > c$. With these quantization thresholds in place, the ingress node simply examines the current link loads and identifies the set of links with the least quantized load. Fig. 2.5 shows a multi-level link state quantizer, where we illustrate two loading conditions in which only the circled links are used in routing the traffic load.

Our DTMP scheme, is aimed at addressing the possible void of links with load less than y^* based on a dynamic threshold mechanism. Instead of routing traffic flows on all links with load less than y^* we route the traffic over all the links that do not have a load exceeding a multiple of the least loaded one. That is, we use all the links with load no more than $(1+\beta) \cdot \min_l x_l(0)$, where $x_l(0)$ is the load on link l at time 0 and β is a positive scaling factor. This guarantees that there is at least one link on which traffic can be routed, *i.e.,* the least-loaded link(s). In the sequel we show by simulation that the DTMP scheme performs well against the other two schemes, especially in the operating regime where the number of links n is modest and the arrival rate at the ingress node is not accurately modeled, *e.g.,* it may vary. We might however expect these routing schemes to be equivalent in the

asymptotic regime considered in Section 2.2.1. By considering the associated asymptotic regimes one can show that β should be approximately set to $\alpha^* \cdot n$. Specifically, we say the two sets of links used by DTMP and DKSP asymptotically "equivalent", if the link loads satisfy the following condition,

$$E[X^n_{([\alpha^* n])}] \leq (1+\beta) \cdot E[X^n_{(1)}] \leq E[X^n_{([\alpha^* n+1])}].$$

As the number of links n and the offered load λn go to infinity $\frac{E[X^n_{([\alpha^* n])}]}{E[X^n_{(1)}]}$ goes to $\alpha^* n$, if the link loads are uniformly distributed a priori. Hence $k \approx \alpha^* n$.

Table 2.1: The selection of β^*.

μ	t	$\beta^* = \alpha^* n$	β^* via simulation
0.25	0.5	3.6	4.5
0.25	1.0	5.4	6
0.1	1.0	8.4	9

Let us assess the performance achieved by setting $\beta^* = \alpha^* \cdot n$. Specifically, we compare these values with the best β^* value obtained via simulation, where a collection of possible values for β^* was examined and that corresponding to the least flow blocking rate was identified. Consider a network with 12 parallel links, each with capacity 20 units. Traffic flows arrive according to a Poisson process with rate equal to 50 flows per second. The flow holding time is exponentially distributed with mean μ^{-1} and each flow requests one unit of bandwidth. We assume a periodic link state update mechanism with period t. Table 2.1 summarizes the comparison across a range of μ and t values. In the simulation β^* was incremented by 0.5 each step in the process of searching for the best value. It is fair to conclude that as a simple approximation, $\beta^* = \alpha^* \cdot n$ provides a crude, but reasonable setting for β^* resulting in good performance.

2.2.2 A comparison of three dynamic multi-path routing schemes

In this section we present our simulations comparing the performance of the three routing schemes discussed above. As a base case, we will use a routing scheme that routes traffic to the least-loaded link. The performance metric we use, is "% routed volume", that is the percentage of the bandwidth demand that is successfully routed. In the sequel we also use " % improved routed volume", which is defined as $\frac{x-y}{y} \cdot 100$, where x is the performance achieved by our dynamic multi-path routing schemes, and y is that achieved by a dynamic single-path scheme. In order to investigate the robustness of the proposed schemes to varying arrival rates we will consider an operational scenario where the network is designed to carry traffic flows with a *nominal* arrival rate, but the *actual* flow arrival rate is different. Note that the performance of the routing schemes depends on the parameters α^*, y^* and β^*. Note that these parameters are all functions of the nominal flow arrival rate λ. Thus we wish to establish the sensitivity of the routing performance to the nominal λ, for the three schemes proposed above.

We first compare the performance of the three routing schemes. We set the parameters α^*, y^*, and β^* based on a nominal flow arrival rate equal to 100 flows per sec, and the actual flow arrival rate vary between 40 to 120 flows per sec. From Figure 2.6 we observe that DTMP scheme exhibits the most significant performance improvement over single path least-loaded routing. In particular, the % improved routed volume for DTMP ranges from 7% to 136%, as the actual flow arrival rate increases.

Next we examine another case where network load was underestimated, *i.e.,* we set the parameters based on a nominal flow arrival rate equal to 10 flows per sec, and let the actual flow arrival rate varies between 40 and 120 flows per sec. From Figure 2.7 We observe that in this case with underestimated operational parameters, DTMP policy again performs adequately, while other schemes see a significant performance degradation. For example, if we compare Figure 2.6 and 2.7 at arrival rate 100 flows per sec, we see that the performance of DTMP remains almost unchanged while those of DKSP and DQSP degrade

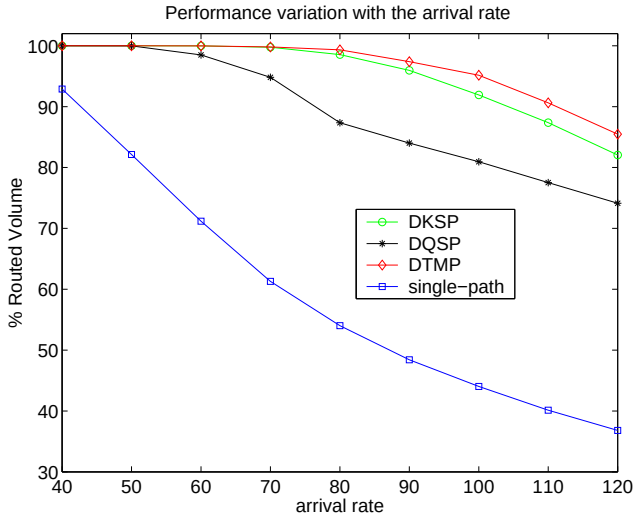

Figure 2.6: Performance comparison of the routing schemes: nominal load = 100.

significantly.

2.3 Dynamic multi-path routing in mesh networks

2.3.1 From parallel links to mesh networks: extending DTMP

In the previous section we considered dynamic multi-path routing problem for a symmetric parallel-link network. It is difficult to extend these results to mesh networks. Specifically, the available routes between a pair of ingress and egress nodes are not necessarily disjoint, so there may be interactions among the traffic loads on various routes. Moreover, there are usually multiple pairs of ingress and egress nodes that make independent routing decisions based on network states, and these decisions may be synchronized, which in turn aggravates congestion in the network.

Let us consider routing a set of traffic flows on a mesh network $G(N, L)$ with a set of nodes N and a set of links L, so that an additive network congestion measure is minimized. Formally, suppose we have a set of ingress-egress node pairs S, a set of available routes R,

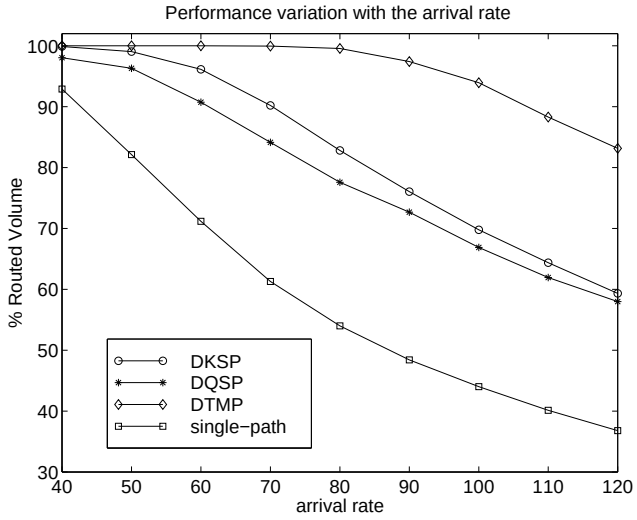

Figure 2.7: Performance comparison of the routing schemes: nominal load = 10.

and a 0-1 matrix H where $H_{sr} = 1$ if $s \in S$ can be served by route $r \in R$, and $H_{sr} = 0$ otherwise. Let $s(r)$ denote the set of routes r that serve flow s, *i.e.*, $H_{sr} = 1$. Moreover, let us define a matrix A such that $A_{jr} = 1$ if route $r \in R$ passes through link $j \in L$. Let us model the network dynamics with a fluid approximation. Suppose the traffic flows arrive with rate g_s, the mean flow holding time μ^{-1} is set to 1, and each flow transmits at unit rate. We define the routing objective as follows:

$$\min_{\lambda_r, r \in R} \sum_{i \in L} \int_0^{x_i} z(y)dy$$
$$\text{s.t.} \quad H\lambda = g, A\lambda = x,$$

where $f(x_i) = \int_0^{x_i} z(y)dy$ is a convex function of x_i, $g = (g_s, s \in S)$ is the vector of the flow arrival rates (or offered load in our setup), and $x = (x_l, l \in L)$ is the vector of the link loads. The solution to this network flow problem can be characterized as follows:

$$\lambda_r > 0 \Rightarrow \sum_{i \in r} z(x_i) \leq \sum_{i \in r'} z(x_i), \forall r' \in s(r).$$

i.e., only the *shortest* paths where link lengths are $z(x_i)$, will carry *positive* amounts of flow. This is known as a Wardrop equilibrium [36]. As a special case, if we were to minimize the

network congestion measure given by $-\sum_{l \in L} \log(c_l - x_l)$, the link cost on a link l with capacity c_l becomes $z(x_l) = 1/(c_l - x_l)$, *i.e.*, the inverse of the available bandwidth. In later sections we will use $1/(c_l - x_l)$ as link metric.

The Wardrop equilibrium suggests that one should route the traffic in such a way that only the "shortest" paths carry positive amounts of flow. However, this is meaningful only in a static or quasi-static network scenario. In the highly dynamic environment we consider in this study, where traffic flows arrive and depart quickly(*e.g.*, less than link state updating period), link loads often exhibit bursty changes, and the link state information used to compute the "shortest" paths is often outdated. Hence the "perfect load-balancing" suggested by the Wardrop equilibrium is neither practical nor achievable[66]. Instead of restricting ourselves to shortest paths alone and trying to adapt to the exact flow proportions, we propose to randomly route the traffic flows between an ingress-egress node pair s, among all the paths with length no more than $(1 + \beta^*) \cdot l_s$, where l_s is the length of the shortest path associated with the node pair s, and $\beta^* > 0$ is a design parameter to be determined.

This approximation is similar to the DTMP scheme proposed for the parallel-link model. We will again use "DTMP" to refer to this approximation scheme for mesh networks. Notice that in the mesh network setup, the set of paths which are selected may not be disjoint, hence some links may be traversed by several paths used for routing the traffic between a given ingress and egress node. The load on these links could "build up." The dynamic aspect of our scheme, *i.e.*, choosing the paths whose length is within a certain range of the shortest path length, helps to avoid this build-up process, as long as the dynamic link metrics, *i.e.*, $1/(c_l - x_l)$ reflect the link loads on the network.

Notice that by letting the length of the paths over which one disperses traffic be dependent upon the shortest path length l_s, we achieve the following intuitive behavior: if the network is lightly loaded, it is beneficial to consistently use only the shortest paths, whose unused capacity is high; if the network is more congested, it is advantageous to spread the load over a larger set of paths in order to accommodate the incoming (bursty)

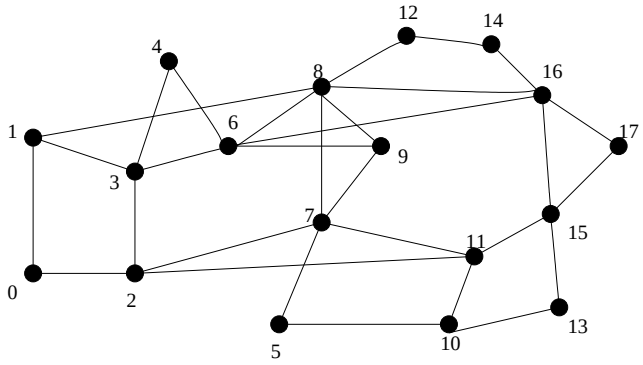

Figure 2.8: NSF topology.

flows. In the next section we examine various aspects of this routing scheme via simulation. Based on our simulations we made the following observations:

1. The DTMP scheme outperforms *dynamic* single-path routing, *i.e.*, least-loaded routing (LLR).

2. In networks with "hot-spots" DTMP offers more significant performance improvement than in networks with "balanced" traffic, thus if such hot spots arise one might resort to DTMP to alleviate the impact of congestion.

3. If traffic flows arrival processes are bursty, DTMP provides a greater performance gain over its single-path counterpart.

4. As the portion of co-located traffic, *i.e.*, traffic between nodes that are one hop away from each other, increases the performance gains from using DTMP decrease.

5. In the network where link state updates are relatively slow as compared to flow arrivals/departures, DTMP offers significant performance improvement.

6. As we scale up the capacity of the network, the use of a DTMP scheme is more important as it offers greater performance gains.

Table 2.2: The traffic matrix.

ingress node	egress node	hop distance	arrival rate
0	16	4	10
1	17	3	10
4	13	4	10
5	14	4	10
8	10	3	10

2.3.2 Simulation setup

We present a set of results for the network shown in Figure 2.8. In the simulation, the flows arrive to the network according to a Poisson process, and the flow holding times are Pareto distributed. The ingress and the egress nodes of the flows are selected according to Table 2.2, which are set up to model a typical WAN traffic pattern, *i.e.*, the ingress and egress nodes of a flow are at least three hops away from each other. In the sequel we will examine the effect of this setup, and evaluate the impact of the "co-located" ingress and egress nodes, *i.e.*, within two hops or less. The parameters for the simulation were set as follows: link capacity is 25 units, mean flow holding time is 1 unit, and the bandwidth request of each flow is uniformly distributed between 0.5 and 1.5 units. This is referred to as the base case. We increase the traffic load by scaling the arrival rates of the base case by a sequence of numbers, shown on the horizontal axis, see Figure 2.9. Unless explicitly stated, we use dynamic link metric $1/(c_l - x_l(t))$, and the routers exchange link states periodically, with an updating period of 0.1 unit.

2.3.3 Performance evaluation

We first compare our DTMP with dynamic single path routing. The performance improvement of the DTMP scheme is evident from Figure 2.9. Specifically, the relative performance improvement ranges from 5% to 13%, as the traffic load grows.

In the above simulation β^* was set to 1.6. In general, it is hard to pin-point the best β^*. It depends on network topology, traffic demands, as well as various timing factors

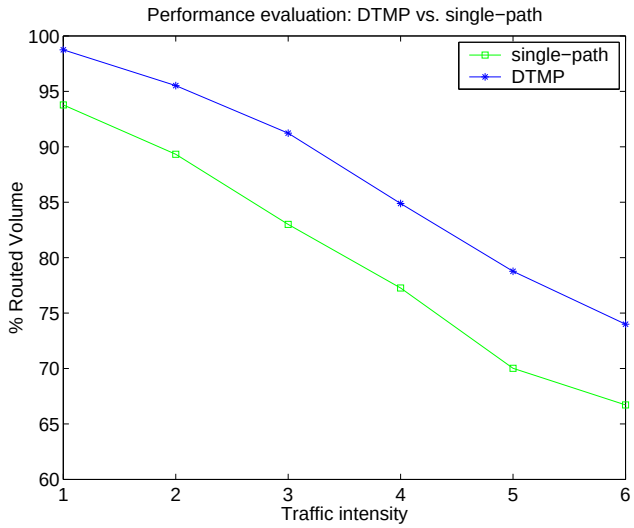

Figure 2.9: DTMP vs. single path.

involved in the network, *e.g.*, flow arrival rate, flow holding time, and link state updating period. Our experience, however, suggests that DTMP's performance is quite robust to the choice of β^*. For the set of simulations we conducted it was observed that in the interval $0.2 \leq \beta^* \leq 3.2$ DTMP outperforms single-path routing. However we did see the performance degradation caused by excessive multi-path routing, in the cases where $\beta^* > 3.2$. We conjecture that in practice it is relatively easy to tune β^* to achieve good overall performance.

From these experiments we conclude that this simple dynamic multi-path routing scheme works well to improve performance over the traditional dynamic single-path routing. The performance of the proposed scheme is relatively robust to the choice of parameter β^*. However, we note that one should not be overly aggressive in setting a high value for this parameter.

To further evaluate the effectiveness of the DTMP scheme, we vary the flow arrivals to the network so that certain "hot-spots" are present. Specifically, we increase the arrival rate from Node 1 to Node 17 to 30 flows per time unit, and decrease the arrival

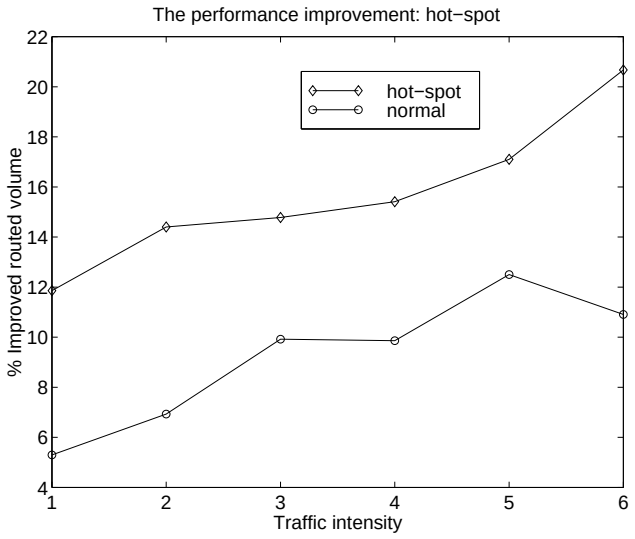

Figure 2.10: The effect of the hot-spot traffic.

rates to other pairs of ingress-egress nodes to 5 flows per time unit. Figure 2.10 compares the performance gain, *i.e.,* the improvement of the performance achieved by DTMP over the single-path routing, with or without the "hot-spot" traffic. A more significant performance improvement (12-21%) is evident when "hot-spots" are present, as compared with the case without "hot-spot" traffic(5-13%). Hence we maintain that DTMP is conducive to alleviating the impact of the "uneven" network loads.

As observed in practice, even traffic flow arrivals themselves may be bursty, *e.g.,*

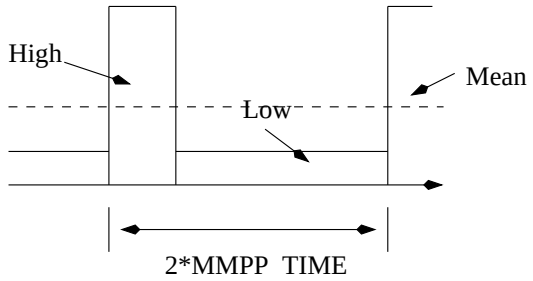

Figure 2.11: Markov Modulated Poisson Process

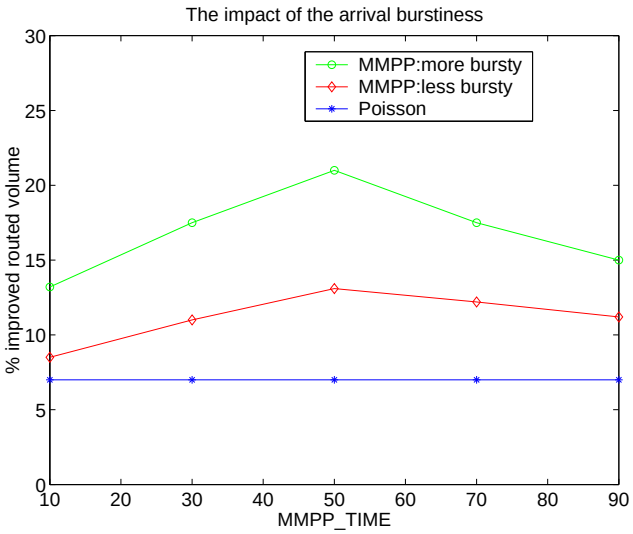

Figure 2.12: The effect of the bursty traffic.

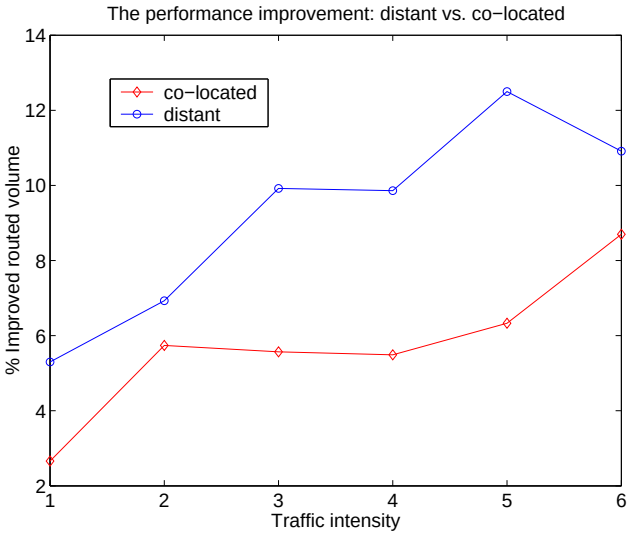

Figure 2.13: The effect of the local traffic.

the access to CNN web site before and after a major news event. We believe that in such an operational scenario, DTMP can deliver more significant performance improvements over its dynamic single-path counterpart. In the previous simulations we modeled the flow arrivals by a Poisson process, which is generally considered a "smooth" random process. To model bursty flow arrivals, we used the Markov Modulated Poisson Process (MMPP) illustrated in Figure 2.11. There are two "modulating" states, "high" and "low". In each state traffic flows arrive as a Poisson process. We will consider two MMPPs with different flow arrival rates in the "high" state. For the first, the flow arrival rate in the "high" state is 3 times the mean given in Table 2.2. For the second, the flow arrival rate in the "high" state is 1.5 times the mean given in Table 2.2. In the "low" state, the traffic flows arrive with rate 1/3 of the mean given in Table 2.2, for both MMPPs. In addition to the rates associated with the "high" and "low" states, the MMPPs are also characterized by the average holding time at "high" and "low" states. For the first MMPP, we set the average holding time at "high" state and "low" state to be $0.5 \cdot$ MMPP_TIME and $1.5 \cdot$ MMPP_TIME, respectively, where MMPP_TIME is a scaling variable which we vary from 10 to 90 units. For the second MMPP, we set the average holding time at both modulating states to be MMPP_TIME.

In Fig. 2.12 we illustrate the performance improvement achieved when such bursty arrival processes are present. It is evident that the DTMP scheme is more effective in a network supporting bursty arrivals processes. In addition, note that when MMPP_TIME equals to 50 units the performance gains are the highest. This suggests an optimal time scale for which DTMP is most effective. The intuition is as follows: when the MMPP_TIME is small, the "high" and "low" states alternate frequently relative to the link state updates, hence the routing decisions that have to be made in the "high" state by the dynamic single path routing scheme "averaged out" with those in the "low" state. If MMPP_TIME is large the network states get updated often enough to track changes in the traffic. The "critical" time scale, however, is the one where a burst of flows arrive in the "high" state and the updates are not quite frequent enough for the single path routing scheme to track such

changes. At this "critical" time scale DTMP provides the most performance improvement over single path routing scheme.

Next let us examine the impact on network performance of traffic locality with respect to the ingress and egress nodes. The traffic arrival pattern in the above simulations roughly models a WAN. One might ask what happens if a significant amount of traffic is between network nodes that are "co-located", *i.e.*, having direct links to each other? Notice that in the topology under consideration, the closer the ingress and egress nodes, the fewer paths there are that have similar characteristics in terms of hop count. Intuitively, this implies that we have fewer *options* over which to support the proposed multi-path routing scheme. Hence we should see a decrease in the performance improvement achieved by DTMP over its single-path counterpart. To verify this intuition, we introduce additional traffic between Nodes 6 and 16, and also between Nodes 7 and 11, each with rate 10 flows per unit, while decreasing the flow arrival rates associated with the other node pairs in Table 2.2 to 6 flows per unit. This is done so that the total flow arrival rate to the network is kept to be 50 flows per unit. The results in Fig. 2.13 support this insight.

The performance of the DTMP routing scheme depends on the quality of the set of paths over which dispersion will take place. The quality of a path is captured by its length, which in turn depends on timely link metrics. In our next simulation we considered how often updates would be generated, namely "slow-updates" and "fast-updates". [3] By "slow-updates" we refer to an operating regime where link metrics are updated every 1 time unit, and by "fast-updates" we refer to an operating regime where link metrics are updated every 0.1 time units. This distinction in link state updating rate may correspond to networks with different geographical coverage, *i.e.*, long versus short-haul networks, or simply limitations on signaling overheads. As seen in Figure 2.14, for "slow-updates" the performance improvement is more significant. The reason is that for "slow-updates", the unevenness and/or buildup in network loads are more pronounced in the single-path routing

[3] We are using a simple periodic updating scheme. Other mechanisms exist and a comparison study can be found in [3].

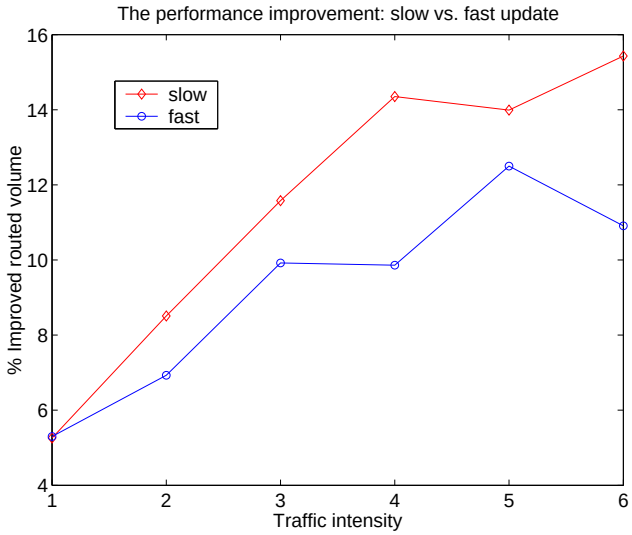

Figure 2.14: The impact of the link state update period.

scheme due to the longer delay in link state update. Hence our DTMP scheme alleviates the impact of delays in distributing link state information.

Finally we examined the impact on the routing performance as the capacity of the network is increased. Let us denote the network used in the previous discussions by NET-SMALL, and construct a new network, NET-BIG, that has the same topology as NET-SMALL but 100 times the link capacity. In order to derive a meaningful comparison, we scaled the flow arrival rates to NET-BIG to be 250 times those of NET-SMALL. A comparison of the performance improvements achieved by DTMP is shown in Figure 2.15.

We observe that performance improvement brought about by the DTMP mechanism are more significant in the network with large capacity. The reason is that with delays in link updating, single-path routing is somewhat oblivious to the network load condition, which leads to poor load balancing on the network. The key point is that such imbalances are more pronounced in the large capacity network and a multi-path routing scheme like ours is able to alleviate this problem more substantially.

Note that in the above simulations we opted to *linearly* scale the flow arrival rate

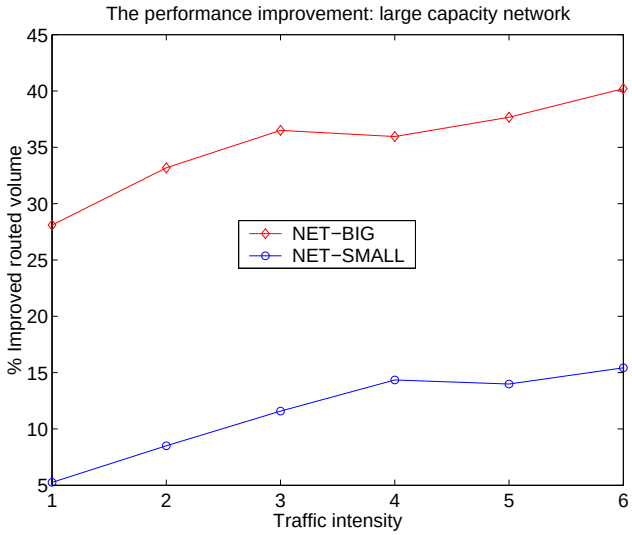

Figure 2.15: The impact of the network capacity.

and the network capacity. To further generalize this result, we also experimented with other scaling schemes. In particular, we scaled the flow arrival rates so that in both networks dynamic *single-path* routing scheme achieved roughly the same performance. We compared the performance improvement attained by DTMP schemes on these two networks and found that in networks with larger arrival rate and link capacity the DTMP scheme again achieved a more significant performance improvement over dynamic single-path routing scheme.

2.4 Summary and discussion

In this chapter we studied dynamic multi-path routing. We formulated a stochastic optimization problem for a parallel-link network model. We analyzed a set of routing policies intended to optimally select the links over which to disperse traffic flows. For an asymptotic regime we exhibited an associated optimization problem which permits a closed-form analysis. These results provide a number of insights addressing the interaction among traffic arrivals, flow holding time, link capacity, and network updating time scales. In particular,

we identified a robust dynamic multi-path routing scheme, *i.e.*, the DTMP scheme, that performs well in various network environments.

We then extend the findings to networks with mesh topologies. We adapted the DTMP scheme in this context and conducted extensive simulations to examine its performance, including the impact of the link state updating rate, burstiness in traffic arrivals, and various issues concerning traffic load distribution. Based on our simulations we believe we have identified a robust dynamic multi-path routing scheme that can be used to effectively route/disperse traffic in high speed networks.

2.5 Appendix

2.5.1 Proof of Theorem 1

To prove Theorem 2.2.1, we will use the following three lemmas.

Lemma 2.5.1 *Suppose $X_1, X_2, \cdots$ are iid uniform random variables on the interval [0,1]. Let $X^n_{(\lceil n\alpha \rceil)}$ be the $\lceil n\alpha \rceil$ order statistic based on the first n random variables in the sequence. Then $X^n_{(\lceil n\alpha \rceil)} \xrightarrow{a.s.} \alpha$.*

Proof : By definition,

$$X^n_{(\lceil n\alpha \rceil)} \xrightarrow{a.s.} \alpha \quad \text{iff} \quad P(\{\omega \in \Omega | \lim_{n\to\infty} X^n_{(\lceil n\alpha \rceil)}(\omega) = \alpha\}) = 1.$$

Now $X^n_{(\lceil n\alpha \rceil)}(\omega) \to \alpha$ if $\forall \epsilon > 0$, $\exists m > 0$, such that $\forall n > m$, $\alpha - \epsilon \le X^n_{(\lceil n\alpha \rceil)}(\omega) \le \alpha + \epsilon$. Note that

$$X^n_{(\lceil n\alpha \rceil)}(\omega) \le \alpha+\epsilon \Longleftrightarrow \sum_{i=1}^{n} 1\{X_i(\omega) \le \alpha+\epsilon\} \ge \lceil n\alpha \rceil \Longleftrightarrow \frac{1}{n}\cdot\sum_{i=1}^{n} 1\{X_i(\omega) \le \alpha+\epsilon\} \ge \frac{\lceil n\alpha \rceil}{n}.$$

By the Strong Law of Large Numbers,

$$\lim_{n\to\infty} \frac{1}{n} \cdot \sum_{i=1}^{n} 1\{X_i(\omega) \le \alpha + \epsilon\} = \alpha + \epsilon \ge \alpha = \lim_{n\to\infty} \frac{\lceil n\alpha \rceil}{n},$$

so

$$\exists m_1, \forall n > m_1, X^n_{(\lceil n\alpha \rceil)}(\omega) \leq \alpha + \epsilon, \forall \omega.$$

Similarly, $\exists m_2, \forall n > m_2, X^n_{(\lceil n\alpha \rceil)}(\omega) \geq \alpha - \epsilon, \forall \omega$. Whence $X^n_{(\lceil n\alpha \rceil)}(\omega) \to \alpha, \forall \omega$ and $X^n_{(\lceil n\alpha \rceil)} \xrightarrow{a.s} \alpha$.

■

Lemma 2.5.2 *If $X_1, X_2, \cdots$ are iid random variables with distribution function F, where F is continuous and has a finite support $[0, c]$, then the order statistics are such that $X^n_{(\lceil n\alpha \rceil)} \xrightarrow{a.s} F^{-1}(\alpha)$.*

Proof: Since $X \sim F(x)$, $F(X) \sim U(0, 1)$. By the continuity assumption, we have that $\forall \epsilon > 0, \quad \exists \epsilon' > 0$,

$$\begin{aligned} &P(\{\omega \in \Omega | \lim_{n\to\infty} X^n_{(\lceil n\alpha \rceil)}(\omega) = \lim_{n\to\infty} F^{-1}(\alpha) \\ &= P(\{\omega \in \Omega | \lim_{n\to\infty} F(X^n_{(\lceil n\alpha \rceil)})(\omega) = \alpha) = 1. \end{aligned}$$

The last step follows from the fact that F is increasing, thus $F(X^n_{(i)})$ is the i-th order statistic of a uniformly distributed random variable $F(X)$. By definition we obtain almost surely convergence.

■

Lemma 2.5.3 *For the continuous function f, if $X \sim F$, then* $\lim_{n\to\infty} \frac{1}{n} \cdot \sum_{i=1}^{\lceil n\alpha \rceil} f(X^n_{(i)}) = E[f(X) \cdot 1\{X \leq F^{-1}(\alpha)\}]$.

Proof:

$$\frac{1}{n} \cdot \sum_{i=1}^{\lceil n\alpha \rceil} f(X^n_{(i)}) = \frac{1}{n} \sum_{i=1}^{n} f(X_i) 1\{X_i \leq X^n_{(\lceil n\alpha \rceil)}\}.$$

Since $X^n_{(\lceil n\alpha \rceil)} \xrightarrow{a.s} F^{-1}(\alpha)$, we have

$$\begin{aligned} &\lim_{n\to\infty} \frac{1}{n} \cdot \sum_{i=1}^{n} f(X_i) \cdot 1\{X_i \leq X^n_{(\lceil n\alpha \rceil)}\} \overset{a.s.}{=} \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} f(X_i) \cdot 1\{X_i \leq F^{-1}(\alpha)\} \\ = \quad & E[f(X) \cdot 1\{X \leq F^{-1}(\alpha)\}], \end{aligned}$$

by the law of large numbers.

■

Proof of Theorem 2.2.1: By Lemma 2.5.3, we have

$$\min_{0\le\alpha\le 1}\{\lim_{n\to\infty}\frac{1}{n}\cdot E[\sum_{i=1}^{\lceil\alpha n\rceil}[f(X^n_{(i)}e^{-\mu t}+\frac{\lambda}{\alpha\mu}(1-e^{-\mu t}))-f(X^n_{(i)})]+\sum_{i=\lceil\alpha n\rceil+1}^{n}[f(X^n_{(i)}e^{-\mu t})-f(X^n_{(i)})]]\}$$

$$\begin{aligned}
&= \min_{0\le\alpha\le 1}\{E[f(X\cdot e^{-\mu t}+\frac{\lambda}{\mu\alpha}(1-e^{-\mu t}))1\{0\le X\le F^{-1}(\alpha)\}]+E[f(X\cdot e^{-\mu t})1\{(X\ge F^{-1}(\alpha)\}]\\
&- E[f(X)1\{(0\le X\le F^{-1}(\alpha)\}]-E[f(X)1\{X\ge F^{-1}(\alpha)\}]\},\\
&= \min_{0\le\alpha\le 1}\{E[f(X\cdot e^{-\mu t}+\frac{\lambda}{\mu\alpha}(1-e^{-\mu t}))1\{0\le X\le F^{-1}(\alpha)\}]+E[f(X\cdot e^{-\mu t})1\{X\ge F^{-1}(\alpha)\}]\\
&- E[f(X)1\{(0\le X\le F^{-1}(\alpha)\}]-E[f(X)1\{X\ge F^{-1}(\alpha)\}]\},\\
&= \min_{0\le\alpha\le 1}\{E[f(X\cdot e^{-\mu t}+\frac{\lambda}{\mu\alpha}(1-e^{-\mu t}))1\{0\le X\le F^{-1}(\alpha)\}]+E[f(X\cdot e^{-\mu t})1\{X\ge F^{-1}(\alpha)\}]\\
&- E[f(X)]\},
\end{aligned}$$

which is equivalent to

$$\min_{0\le\alpha\le 1}E[f(X\cdot e^{-\mu t}+\frac{\lambda}{\mu\alpha}(1-e^{-\mu t}));0\le X\le F^{-1}(\alpha)]+E[f(X\cdot e^{-\mu t});X\ge F^{-1}(\alpha)]$$

By a change of variable, $\alpha=F(y)$, we have that (2.5.1) is equivalent to

$$\min_{0\le y\le c}E[f(X\cdot e^{-\mu t}+\frac{\lambda}{\mu F(y)}(1-e^{-\mu t}));0\le X\le y]+E[f(X\cdot e^{-\mu t});X\ge y]$$

■

2.5.2 Proof of Fact 1

Proof: If $X\sim$ Uniform$[0,c]$, the first order optimality condition for (2.6) is given by:

$$f(ye^{-\mu t}+\frac{\lambda c}{\mu y}\cdot(1-e^{-\mu t}))-f(ye^{-\mu t})=\frac{\lambda c\cdot(1-e^{-\mu t})}{y^2\cdot\mu\cdot e^{-\mu t}}\cdot[f(ye^{-\mu t}+\frac{\lambda c}{\mu y}\cdot(1-e^{-\mu t}))-f(\frac{\lambda c}{\mu y}\cdot(1-e^{-\mu t}))].$$

Let $\eta=e^{-\mu t}$ and $\theta=\frac{\lambda c}{\mu}\cdot(1-e^{-\mu t})$, then we can write this condition as

$$f(\eta y+\frac{\theta}{y})-f(\eta y)=\frac{\theta}{\eta y^2}\cdot[f(\eta y+\frac{\theta}{y})-f(\frac{\theta}{y})],$$

or equivalently,

$$\frac{f(\eta y + \frac{\theta}{y}) - f(\eta y)}{\frac{\theta}{y}} = \frac{f(\eta y + \frac{\theta}{y}) - f(\frac{\theta}{y})}{\eta y}, \tag{2.7}$$

since we assume f is convex it follows that there exists a unique solution y^+ to (2.7) and $\eta y^+ = \frac{\theta}{y^+}$, or equivalently, $y^+ = \sqrt{\frac{\theta}{\eta}}$.

Now the optimizer y^* is either the stationary point y^+ or a boundary point 0 or c. Note that

$$E[f(X \cdot e^{-\mu t} + \frac{\lambda c}{\mu y}(1 - e^{-\mu t}); 0 \leq X \leq y)] \to \infty$$

as $y \to 0$, so $y^* = \min\{y^+, c\}$ and $\alpha^* = y^*/c$.

■

Chapter 3

Predictive Routing To Enhance QoS For Stream-based Flows Sharing Excess Bandwidth

3.1 Introduction

We investigate routing mechanisms for stream-based traffic flows. The traffic and service model we consider can be summarized as follows: upon arrival to the network the traffic flows require a minimal level of guaranteed service, *e.g.,* in terms of a minimal bandwidth guarantee. The flow is admitted if there are sufficient resources to do so along the selected route, otherwise the flow is rejected. After admission into the network, the flows[1] may achieve improved performance by sharing network resources which are not in use to guarantee service for ongoing flows. In general, the performance achieved by a given flow depends on the resource allocation policy at the flow level and the packet scheduling policy. At the flow level, the shared resources are allocated to ongoing flows based on the resource sharing policies employed in the network, *e.g.,* TCP[33] or max-min sharing[11]. At the packet level, the packet scheduling policy determines the packet service rate for a given flow in accordance with the flow level resource allocations. In this chapter we focus on analyzing routing schemes that improve the overall network performance at the flow level. We believe such improvements in flow level performance, coupled with a suitable

[1] In this chapter we refer to traffic flows and their associated users interchangeably.

work-conserving scheduling policy, can lead to better user-perceived QoS.

For a given resource sharing policy, the performance achieved by a given flow during its sojourn in the network depends on a number of factors including the number of flows in the network, the resources available for sharing, and the set of links traversed by these flows, *i.e.,* their routes. This motivates us to investigate routing mechanisms that not only optimize system metrics such as the *flow blocking rate,* but also enhance the *user-perceived* performance to the individual flows. Prominent service classes that fit in this generic service model include ATM VBR service[19] and rate adaptive applications[58]. Specifically, ATM VBR connections would request a level of QoS, *e.g.,* cell loss rate, upon arrival to the network. The Call Admission Control (CAC) mechanism employed by the network might then translate this user-centric QoS specification into an estimate of the resources required to satisfy the user QoS demand, *e.g., effective bandwidth*[37]. Given an estimate for the effective bandwidth the network decides to admit or reject connections. Note that from a user's perspective, it is beneficial to route the admitted VBR connections on a path with additional spare resources, so that the inaccuracies in the estimates of the effective bandwidth can be better tolerated. In the case of rate adaptive applications, traffic flows arriving to the network are given a minimal bandwidth guarantee, and expect variable transmission rates, *i.e.,* when the load is lower (higher), the flows adapt to higher (lower) transmission rates, possibly by subscribing (unsubscribing) to additional service layers [48]. In this case the excess bandwidth seen by the flows might be used to support lower priority layers. These observations suggest that it might be beneficial to route these flows so as to minimize the average load a flow is likely to see during its sojourn in the network.[2] We shall refer to the average load seen by a flow as the *flow-perceived load*, and set out to design a routing scheme that aims at improving this performance measure, in addition to minimizing the flow blocking rate.

To achieve this goal, we consider routing schemes that use prior knowledge of the

[2]Equivalently, we might attempt to maximize the average available bandwidth a flow is likely to see during its sojourn in the network.

flow holding time. For example, the holding time might be known or characterized via its mean or distribution. We propose to model the link load dynamics as a means to estimate the expected flow-perceived load on the network links. As in [8], where a call admission control scheme is studied, we will use the queuing-theoretic results in [43] to propose a parametric model for the link load dynamics. In our routing framework, links estimate and advertise the parameters associated with their loads in addition to their current states. New flows are routed based on this information and prior knowledge of their holding times so as to minimize the expected flow-perceived load. We will show that even with limited information on flow holding times, *e.g.,* their means, one can often improve both the flow blocking rate and flow-perceived load[3] *simultaneously*. Considering that the improved flow blocking rate implies an increase in the load supported by the network, it is remarkable that one can also achieve better performance in terms of flow-perceived load, and thus better eventual QoS. To substantiate this claim, we will show that in a network where available bandwidth is shared in fair fashion, a significant increase in a flow's bandwidth share can be realized when using our routing approach versus two baseline schemes.

In order to study the effectiveness of our approach, we examined various operational issues by simulations. We believe the proposed routing scheme operates effectively in a wide range of contexts, and its performance is robust to various uncertainties in the network's operating environments.

3.1.1 Related work

As mentioned above, in this chapter we propose a routing scheme that routes traffic flows based on both link load dynamics and prior knowledge on flow holding time. We will use an auto-regressive process to model the link load dynamics, and estimate its parameters based on load samples. The key idea is to integrate such information in the notion of the *expected flow-perceived load*, and route the traffic flows so that the expected load seen by

[3]The flow-perceived load is measured by averaging individual flows's perceived load over all flows that are served by the network.

flows during their sojourn in the network is minimized. Our work contributes to ongoing research on routing QoS traffic and work considering the role that prior knowledge of flow holding times might play.

Specifically, in [53] a number of competitive routing algorithms are presented for ATM networks. The results indicate that one can design online routing algorithms to achieve different degrees of competitiveness with respect to the optimal offline algorithm, depending on the assumptions made concerning prior knowledge of connection holding times. Rather than focusing on designing a good routing scheme relative to the worst case arrival process, in this chapter we optimistically assume that link loads follow quasi-stationary stochastic dynamics.

In [57], a routing scheme is proposed which provides differentiated handling of short versus long-lived flows. Data packets are routed on static shortest paths, until a flow classifier is triggered to switch the flow routing based on a dynamic algorithm that is load-sensitive. Our work differs from [57] in that we use dynamic routing for all the traffic flows, but the differentiation is done through the use of different routing metrics for different flows. Instead of relying on a flow classification trigger as in [57], our scheme explicitly determines per-flow routing behavior by integrating into the routing decision the (mean) flow holding time and the estimated parameters characterizing link load dynamics.

In [42] a number of routing algorithms are examined in a network where bandwidth is shared among best effort traffic flows according to the max-min fair criterion. The authors propose a routing metric which approximately estimates the max-min rate for the new connection upon arrival. The resulting shortest path algorithm outperforms minimum hop routing and shortest-widest path routing in terms of packet throughput. Our work differs from [42] in several aspects: (1) we focus on improving the performance (*i.e.*, blocking *and* flow bandwidth share) of stream-based flows instead of max-min rate share of the best-effort file transfers; (2) we use a link state information that scales better than that used in [42], where each link needs to maintain a sufficient number of "rate scales" in order to obtain

an adequate estimate for the rate share of the new connection, and (3) we believe that the notion of expected flow-perceived load effectively captures the resource-sharing potential in a network, thus routing schemes incorporating this notion will apply to other resource sharing criteria such as proportional fair share [23, 45] and size-based sharing[72].

More generally, there has been extensive study of dynamic routing, *e.g.*, on its instability if done at the packet level [67], or on approaches to minimize blocking rate at the flow level by ensuring a better "load-balancing" [3, 41, 39, 36, 22, 1, 68, 51, 24]. Our approach can also be categorized as a load-balancing scheme. However, it differs from the afore-mentioned work, not only in terms of the specific routing metrics we propose, but also in that as a routing objective we explicitly identify the improvement of the individual flow's perspective of resource sharing potential.

In the following sections we present asymptotic approximations for the link load dynamics and the associated parameter estimation techniques, based on which our routing algorithm is constructed. We then examine various factors related to this routing scheme, propose an extension to mesh networks, and discuss simulation results that validate the effectiveness of our approach.

3.2 Analysis: a simple parallel-link model

Let us consider a simple parallel link model, where a source node s and a destination node d are connected by n links, see Fig.3.1. Each link i has a capacity of c units, and serves an exogenous flow load which arrives according to a Poisson process with rate λ_i. Each flow has an exponentially distributed holding time with mean μ_i^{-1}, and requires one unit of bandwidth to ensure its minimal QoS guarantee. In this chapter we will model the link load dynamics associated with the *minimal* bandwidth commitments the network has made and make routing decisions based on this model to improve the QoS of flows sharing *excess* (or additionally available) bandwidth. We denote the number of flows in progress on link i, or equivalently the load at time t, by $X_i^c(t)$, where the superscript c indicates that this is

the load process on a link with capacity c. We will subsequently consider two asymptotic regimes where c and λ_i grow.

3.2.1 A new routing metric: expected flow-perceived load

We consider routing flows that arrive at node s and are destined to node d. Let us assume the considered flow load from node s to node d is *small* in comparison with the load generated by the exogenous flow processes described above, so that the routing of the flows from s to d will not affect the stationary link loads $X_i^c(t)$, $i = 1, 2, \cdots n$. Suppose a single flow with a known holding time h is to be routed at time 0 from node s to node d. Let link loads be $X_i^c(0) = x_i^c(0)$, $i = 1, 2, \cdots n$. As discussed in the introduction we propose to route the flow to the link where it is likely to experience a minimal load during its sojourn in the network.

We define the *flow-perceived load* as the time average of the load during the flow's sojourn in the network. Thus suppose a new flow is to be routed at time 0, we can express the *expected flow-perceived load* on link i as

$$\begin{aligned} u_i(h, x_i^c(0)) &:= E[\frac{1}{h}\int_0^h X_i^c(t)dt | X_i^c(0) = x_i^c(0)] \\ &= \frac{1}{h}\int_0^h E[X_i^c(t)|X_i^c(0) = x_i^c(0)]dt. \end{aligned}$$

As mentioned in the introduction this metric quantifies the expected load a flow with known holding time h would see on link i. We propose to route the flow to the link i with maximum *expected flow-perceived available bandwidth, i.e.,* $c - u_i(h, x_i^c(0))$. When all links have the same capacity this is equivalent to minimizing the expected flow-perceived load $u_i(h, x_i^c(0))$. We will later relax the assumption that h is known and examine the sensitivity of such routing algorithms to the knowledge of the flow holding time. Below we consider some approximations for the expected flow-perceived load, assuming the link load dynamics are independent of the routing decisions.

3.2.2 First approximation: a fluid model

Consider an asymptotic regime where λ_i and c approach infinity, but $\lambda = \theta_i \cdot c$, *i.e.*, the flow arrival rate increases linearly as link capacity increases, irrespective of the link load condition. As shown in [43], it follows that $\frac{X_i^c(t)}{c} \xrightarrow{a.s.} x_i(t)$ as $c \to \infty$, for $0 \leq t \leq s$, $\forall s < \infty$, where $\{x_i(t)\}$ satisfies the following ordinary differential equation

$$\dot{x}_i(t) = \theta_i - \mu_i x_i(t).$$

Thus if $\frac{X_i^c(0)}{c} \xrightarrow{a.s.} x_i(0)$, we have that

$$x_i(t) = x_i(0) \cdot e^{-\mu_i t} + \frac{\theta_i}{\mu_i}(1 - e^{-\mu_i t}). \tag{3.1}$$

Hence for a link with large capacity c and such that $X_i^c(0) = x_i^c(0)$ we have that roughly

$$X_i^c(t) \approx x_i^c(0) \cdot e^{-\mu_i t} + \frac{\lambda_i}{\mu_i}(1 - e^{-\mu_i t}). \tag{3.2}$$

Using this asymptotic regime we can approximate the expected flow-perceived load introduced in the previous subsection as follows. Assume the link has a large capacity c and its load is $X_i^c(0) = x_i^c(0)$ then by (3.2) we have that

$$\begin{aligned} u_i(h, x_i^c(0)) &\approx \lim_{c\to\infty} E[\frac{1}{h}\int_0^h X_i^c(t) \cdot dt | X_i^c(0) = x_i^c(0)] \\ &= (x_i^c(0) - \frac{\lambda_i}{\mu_i}) \cdot \frac{1 - e^{-\mu_i h}}{\mu_i h} + \frac{\lambda_i}{\mu_i}. \end{aligned}$$

Observe that for short flow holding times the expected flow-perceived load corresponds to the link's state $x_i^c(0)$, and for long flow holding times the expected flow-perceived load tends to the long-term average load $\frac{\lambda_i}{\mu_i}$. Hence if $x_i^c(0) < \frac{\lambda_i}{\mu_i}$, *i.e.*, the initial load is lower than the long-term average load, flows with short holding times will see a lower expected flow-perceived load than those with longer holding times. Conversely, if the initial load is higher than the long-term average load, flows with longer holding times will see a lower expected flow-perceived load than those with shorter holding times.

Fig.3.2 illustrates a special case with two links between source node s and destination node d. The incoming flow may encounter a number of situations with different initial

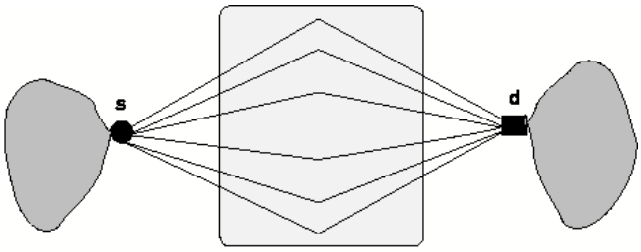

Figure 3.1: A simple parallel-link topology.

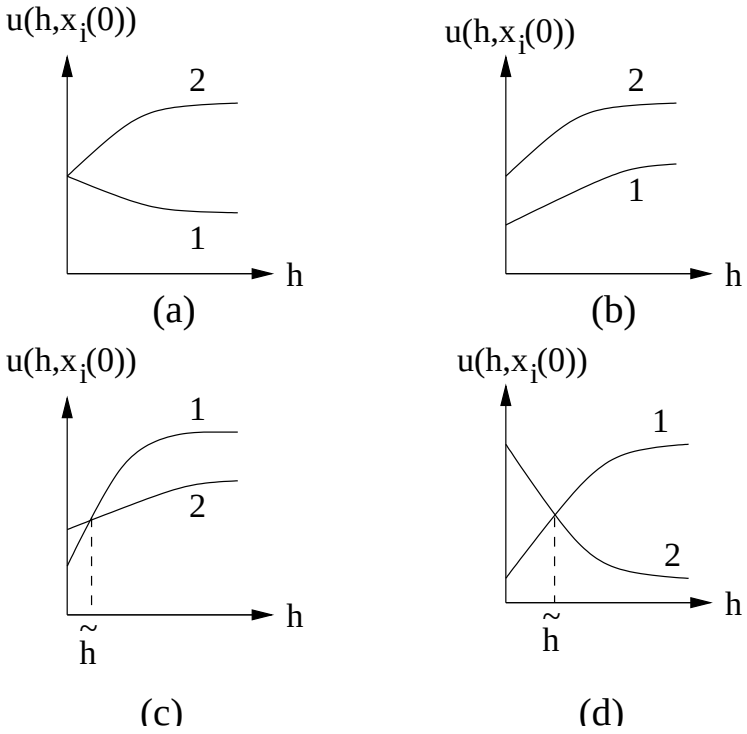

Figure 3.2: Routing in simple topology.

link loads and long-term average link loads. Specifically, for Case (a), Link 1 is preferred even though the initial link loads at time 0 are the same for the two links. For Case (b), Link 1 is preferred since both its initial load and long-term average load are lower than those of Link 2. For Case (c), there exists a "cross-over" flow holding time $\tilde{h}$ where

$$u_1(\tilde{h}, x_1^c(0)) = u_2(\tilde{h}, x_2^c(0)).$$

For flows with holding time shorter than $\tilde{h}$ Link 1 is preferred, and for flows with holding time longer than $\tilde{h}$ Link 2 is preferred. For Case (d), there exists a "cross-over" flow holding time where for flows with holding time shorter than $\tilde{h}$ Link 1 is preferred, and for flows with holding time longer than $\tilde{h}$ Link 2 is preferred. These cases exemplify the potential gains that can be achieved by judiciously accounting for both the flow holding time and link load dynamics.

3.2.3 Second approximation: a diffusion model

The fluid model presented in the previous section allows us to approximately characterize the evolution of the link load dynamics. This model arises when we examine the scaled link load $X_i^c(t)/c$ in the limiting regime where the link capacity c and load $\lambda_i = \theta_i \cdot c$ grow linearly. We can also establish a similar relationship characterizing link load dynamics by investigating the scaled stochastic fluctuations of the link load about its mean.

Suppose a "mode" exists for the limiting regime, *i.e.*, $\frac{X_i^c(t)}{c} \xrightarrow{a.s.} x_i(t) = \frac{\theta_i}{\mu_i}$, then as proven in [43] as $c \to \infty$ the fluctuation process about the mode converges to an Ornstein-Uhlenbeck process. In particular as $c \to \infty$, $\frac{X_i^c(t) - c\theta_i/\mu_i}{\sqrt{c}} \xrightarrow{dist.} X_i(t)$, where $X_i(t)$ satisfies the following stochastic differential equation

$$dX_i(t) = -\mu_i X_i(t) + \sqrt{2\theta_i} dB_i(t),$$

where $\{B_i(t)\}$ is a standard Brownian motion. Thus we can approximately model the link load process as an Ornstein-Uhlenbeck process, which is the solution to the following

stochastic differential equation

$$dX_i^c(t) = -\alpha_i(X_i^c(t) - \rho_i)dt + \sigma_i dB(t), \tag{3.3}$$

where $\alpha_i = \mu_i$, $\rho_i = c\theta_i/\mu_i$, $\sigma_i = \sqrt{2c\theta_i}$.

Consider again a flow with holding time h to be routed to a link i with $X_i^c(0) = x_i^c(0)$ and whose load dynamics are characterized by the above Ornstein-Uhlenbeck process. The expected flow-perceived load in this regime would be given by

$$\begin{aligned} u_i(h, x_i^c(0)) &\approx \frac{1}{h}\int_0^h E[X_i^c(t)|X_i^c(0) = x_i^c(0)]dt \\ &= \frac{1}{h}\int_0^h [(x_i^c(0) - \rho_i)e^{-\alpha_i t} + \rho_i]dt \\ &= \rho_i + (x_i^c(0) - \rho_i)\frac{1 - e^{-\alpha_i h}}{\alpha_i h}. \end{aligned} \tag{3.4}$$

This is similar to the expected flow-perceived load obtained for the fluid model, even though in this case the load dynamics are modeled by stochastic fluctuations about the mode. The reason for this similarity lies in the fact that we are focusing on the mean of the link load process versus the second order statistics inherent in the diffusion approximation. Indeed, the proposed routing metric does not depend explicitly on σ_i, and thus, in a sense, does not capture the degree of fluctuation in the perceived load a flow might see. However, as shown in the sequel, using only the first order characteristics for the perceived loads already achieves significant performance gains. The impact of the second order statistics on the resulting QoS seen by flows is left for future study. In the following sections we will use (3.4) as our link metric and will assume the load process can be adequately modeled by an Ornstein-Uhlenbeck process.

3.2.4 Link load characteristics: parameter estimation

In order to make routing decisions based on the proposed link metric we will estimate the parameters (i.e., ρ_i, α_i) for the load process model for each link. Note that in practice the flow arrivals seen by a link would not be Poisson with a constant rate, as assumed above.

Instead the arrival rates are likely to depend on the current state of the network, *i.e.,* if the link load is low, one might expect to see a higher arrival rate, and if the link load is high the arrival rate might go down. However, in general the dynamics of this process will exhibit the "mean reversion" property of the Ornstein-Uhlenbeck process, *i.e.,* there exists a "mode", and the link load exhibits fluctuations about this mode due to arrivals to, and departures from the system. These in turn are influenced by the routing decisions that are being made.

Let us thus consider modeling the link load process $\{X_i(t)\}$ associated with link i as an Ornstein-Uhlenbeck process with parameters $(\rho_i, \alpha_i, \sigma_i)$. To estimate the needed parameters we sample the link loads every Δ time units. Define the sampled process $y_i(k) = X_i(k\Delta)$ for $k \in Z$. The parameters can be estimated using the following:

$$\hat{\rho}_i = \frac{1}{n}\sum_{k=1}^{n} y_i(k), \quad \hat{\alpha}_i = -\frac{\ln \hat{\beta}_i}{\Delta},$$

where

$$\hat{\beta}_i = \frac{\sum_{k=2}^{n}(y_i(k) - \hat{\rho}_i)(y_i(k-1) - \hat{\rho}_i)}{\sum_{k=1}^{n}(y_i(k) - \hat{\rho}_i)^2},$$

For completeness a detailed derivation is included in the appendix.

Note that the selection of sampling period Δ and sampling window n impact the quality of the parameter estimates. In the sequel we use simulations to assess the importance of these sampling parameters. It is known that the spectrum of the Ornstein-Uhlenbeck process is of the "low-pass" type, *i.e.,* with a cut-off frequency (3dB point) at α_i, hence one might roughly argue that the sampling rate should be at least $2\alpha_i$. For the queuing models discussed earlier the cutoff frequency α_i equals to μ_i, *i.e.,* the flow departure rate. However, in practice the routing mechanism itself would accelerate the mean reversion thus one should expect to require a sampling rate faster than $2\mu_i$, *i.e.,* $\Delta < \frac{1}{2\mu_i}$.

3.2.5 Dynamic or adaptive routing?

Routing algorithms are often said to be either *dynamic*, *i.e.,* using most up-to-date link states, or *adaptive, i.e.,* using averaged/filtered link states. The proposed routing metric is based on *both* the most up to date link states *and* the averaged parameters quantifying the "stationary" or long-term characteristics of the link loads. As observed earlier as the flow holding time h becomes small the proposed metric is essentially a dynamic one, *i.e.,* the current link state, while for large h the longer term characteristics of the link's load are used to make the routing decisions.

To be precise consider a link whose load dynamics is characterized by parameters (ρ_i, α_i). In this case the link load relaxes exponentially to the long term average ρ_i, see (3.1) and (3.2). The "relaxation time" is roughly $1/\alpha_i$. By contrast the expected flow-perceived load is defined as the expectation of the time-averaged link load, and thus relaxes more slowly. Its effective "relaxation time" is roughly e/α_i. Let $h_i^r = e/\alpha_i$. Thus for sufficiently large holding times, *i.e.,* $h > h_i^r$, we have that $u_i(h, x_i(0)) \approx \rho_i$ and the routing of a flow with such holding times may be said to be adaptive. By contrast if its holding time is smaller than h_i^r one might say the metric accounts for the dynamic characteristics of the link's load.

For our simple topology with 2 links, let h_i^r corresponds to the "critical" flow holding time for link i, where $i = 1, 2$. We observe that for all flows with flow holding times greater than $\max\{h_1^r, h_2^r\}$, the routing mechanism is adaptive. Similarly, for all flows with holding time less than $\min\{h_1^r, h_2^r\}$ the routing mechanism is essentially a dynamic one.

In summary, these criteria roughly show a "split" between flows with different holding times, according to which flows are routed in a dynamic or adaptive manner.

3.2.6 Impact of the delays in advertising link states

In a link-state routing scheme, there usually exists a broadcasting mechanism through which the link states at the routers are updated. Inevitably updating delays are involved in such

broadcasting schemes, due to overhead constraints on message propagation and processing delays. In this subsection we examine the impact of updating delays on the proposed routing metrics. Consider the scenario where we make a routing decision at time t, but only have access to the advertised link state at time $t-d$. Without loss of generality suppose $t=d$ and at that time we have access to $x_i(0)$ as well as the parameters (α_i, ρ_i) characterizing the link load. If the delay d is known one can compensate for this by computing the expected flow-perceived load as follows,

$$\begin{aligned} u_i(h, x_i(0); d) &= \frac{1}{h}\int_d^{h+d} E[X_i(s)|X_i(0)=x_i(0)]ds \\ &= \rho_i + (x_i(0)-\rho_i)(\frac{1-e^{-\alpha_i h}}{\alpha_i h})e^{-\alpha_i d} \\ &\approx \rho_i + (x_i(d)-\rho_i)(\frac{1-e^{-\alpha_i h}}{\alpha_i h}) = u_i(h, x_i(d)), \end{aligned}$$

since $x_i(d) \approx (x_i(0)-\rho_i)e^{-\alpha_i d}+\rho_i$. We observe that as d increases $u_i(h, x_i(0); d)$ converges to ρ_i. Thus if significant delays are involved in link-state updates, the routing algorithm that accounts for the (known) updating delays would be essentially adaptive.

Note that this discussion assumes that the delay associated with the current update for the link state is known. In practice this can be done by time-stamping link state updates. However, in the sequel we will, for the most part, not assume such delays are known. Instead, outdated link states are treated as "current" and directly used in estimating the expected flow-perceived load according to (3.4), *i.e.,* when making routing decisions at time d we use $x_i(0)$ in place of $x_i(d)$. Let $\tilde{x}_i(d) = x_i(0)$. In this case

$$\begin{aligned} u_i(h, \tilde{x}_i(d)) &= \rho_i + (x_i(0)-\rho_i)(\frac{1-e^{-\alpha_i h}}{\alpha_i h}) \\ &\approx \rho_i + (x_i(d)-\rho_i)(\frac{1-e^{-\alpha_i h}}{\alpha_i h})e^{\alpha_i d}. \end{aligned}$$

Hence if $d \ll \alpha_i^{-1}$, $u_i(h, \tilde{x}_i(d)) \approx u_i(h, x_i(d))$. We will see in the sequel that even in the case where $d \approx \alpha_i^{-1}$ the predictive flow-time aware routing scheme still provides performance improvements over our baseline schemes. However, the "time-stamping" mechanism can contribute to additional performance improvements.

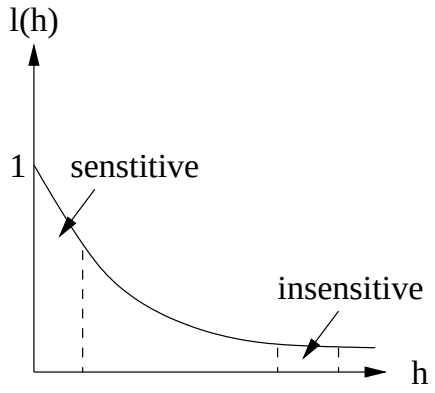

Figure 3.3: Sensitivity to flow holding time.

3.2.7 Uncertainty in flow holding times

Previously we assumed that flow holding times were known in advance. In practice this may not be the case. In this subsection we consider the sensitivity of the routing decisions to uncertainty in the flow holding time. We approach this via two different avenues, (1) what is the impact of uncertainty in the flow holding time on the *routing metric*, *i.e.*, the expected flow-perceived load?, and (2), when do the *routing decisions* change as flow holding times vary? We shall write the expected flow perceived load on link i as

$$u_i(h, x_i^c(0)) = \rho_i + (x_i(0) - \rho_i) l_i(h),$$

where $l_i(h) = \frac{1-e^{-\alpha_i h}}{\alpha_i h}$ and h denotes a known holding time. Note that $l_i(h)$ is decreasing and convex in h, thus the proposed routing metric is fairly insensitive to the uncertainty in h when h is large. Suppose only the mean $\bar{h}$ of a flow's holding time distribution is known. Let H be a random variable with that distribution. One might consider using $u_i(\bar{h}, x_i(0))$ as a routing metric. We note that if $\bar{h}$ is large and the variance of H is small then this metric is fairly representative of the *actual* expected flow-perceived load.

Even if h is moderate or small and so that $l(h)$ is relatively sensitive to h, we argue that although the routing metric $u_i(h, x_i(0))$ may vary if we use $\bar{h}$ instead of the actual flow holding time, the routing decisions based on this may not. Fig.3.2 provides an illustration.

For Fig.3.2-(a-b), we observe that no matter how h varies, the routing decisions remain the same. For Fig.3.2-(c-d), there exists a certain timescale $\tilde{h}$ such that for all h less than $\tilde{h}$ Link 1 is favored, and for all h greater than $\tilde{h}$ Link 2 is favored. Hence in these cases if h and $\bar{h}$ remain on the same side of $\tilde{h}$, the routing decisions made based on $\bar{h}$ will not change.

3.3 Predictive flow-time aware routing in a mesh network

In the previous section we proposed a routing scheme based on the notion of *expected flow-perceived load*, in the context of a simple parallel-link topology. The basic ideas generalize to mesh networks with multiple source-destination pairs routing flows simultaneously. Note that in this case the *link metrics* must be used to construct *path metrics*. The task here is to compute paths for the incoming traffic flows so that (1) the network can carry as many traffic flows as possible, and (2) the perceived loads by the admitted flows during their sojourn in the network are as low as possible. To achieve these goals, we will have to make a number of design choices:

- whether to use additive or concave link metrics to construct path metrics;
- how to incorporate the notion of expected flow-perceived load into the link metrics;
- how to effectively estimate the parameters that characterize the link load dynamics and the expected flow-perceived load.

To systematically address these issues, we have performed extensive simulations of the proposed routing approach. Below we show the performance of our predictive flow-time-aware routing scheme, and illuminate a number of factors that may impact its performance.

3.3.1 Simulation setup

We performed simulations for different network topologies and traffic matrices. In the following we present a set of results for the network shown in Fig.2.8. In our simulations,

Table 3.1: The traffic matrix.

ingress node	egress node	hop distance	arrival rate
0	16	4	50
1	17	3	50
2	16	3	50
2	13	3	50
3	9	2	50
4	13	4	50
5	14	4	50
8	10	3	50
10	4	5	50
11	4	4	50

the flows arrive to the network according to a Poisson process, and the flow holding times are randomly distributed. We experimented with various flow holding time distributions, *e.g.*, exponential, Pareto, hyper-exponential, bi-modal. The general trends of the results are similar under different holding time distributions. We will only show results corresponding to the exponential distributions. The ingress and the egress nodes for the new flows are selected according to Table 3.1, which corresponds to a typical WAN traffic pattern, *i.e.*, the ingress and egress nodes of a flow are at least two hops away from each other.

The parameters for the simulation were set as follows: link capacity is 200 bandwidth units. The mean flow holding time is 1 time unit, which might represent, say a 3-minute period for voice applications, or a 1 hour period for video transmissions. In the following simulations we will use the mean instead of the exact value of flow holding time to evaluate the expected flow-perceived load. The various timescales we will encounter in this section, *e.g.*, link load sampling period, sampling window size, and link state updating delays, will all be set relative to the mean flow holding time. The bandwidth request of each flow is uniformly distributed between 0.5 and 1.5 bandwidth units. This setup is referred to as the base case. We increase the traffic load by scaling the arrival rates of the base case by a sequence of factors. The links in the network estimate the parameters that characterize their load dynamics, *i.e.*, ρ_i, α_i, and distribute these parameters along with the current link load periodically. We will refer to our routing scheme FTAR (Flow Time Aware Routing).

Unless stated explicitly, the current link states are assumed to be known, *i.e.,* no updating delays. We will compare FTAR with two baseline routing schemes. The first is referred to as DSP (Dynamic Single Path), and uses the reciprocal of the *current* available capacity as the routing metric [41]. The second baseline scheme is referred to as MSP (Mean Single Path) and uses the reciprocal of the *estimated mean* available capacity as the routing metric, *i.e.,* $c - \rho_i$. Here ρ_i is the mean load estimated by FTAR. FTAR is a revised version of DSP, *i.e.,* we use expected flow-perceived load instead of current link load to evaluate the available capacity. For an incoming flow, we compute the shortest path based on the inverse of the expected flow-perceived bandwidth, *i.e.,* the link capacity minus the expected flow-perceived load, and establish the flow on the resulting path if the available bandwidth along the path allows it. Otherwise the flow is blocked.

We compare routing schemes based on the *percentage of the bandwidth demands* that are successfully routed, and the *average flow-perceived excess bandwidth* seen by flows. The latter is determined by first sampling the residual bandwidth seen by a given flow during its sojourn in the network, then averaging these samples to get its *perceived excess bandwidth* when it departs, and finally averaging over all the departed flows. Note that this is a crude measure of how much bandwidth there is in the network for a given flow to share with other ongoing flows during its sojourn, *i.e.,* the *potential* for better performance, but not necessarily the bandwidth *achieved* by the flow. Clearly, the *flow-achieved bandwidth* depends on the specific bandwidth sharing policy used in the network, *e.g.,* max-min sharing, proportional sharing[45], or size-based bandwidth sharing[72]. In the sequel we use (weighted) max-min sharing to illustrate the effectiveness of our routing scheme in terms of flow-achieved bandwidth.

Moreover, in the following sections we will evaluate "% improved routed volume" and "% improved average flow-perceived bandwidth", which are defined as $\frac{x-y}{y} \cdot 100$, where x is the performance (% routed volume or average flow-perceived bandwidth) achieved by our FTAR scheme, and y is that achieved by the corresponding baseline scheme.

3.3.2 Parameter estimation: the optimal sampling rate and window size

Let us first examine the impact on the routing performance of the parameter estimation procedure. In particular, we focus on determining a good choice for the sampling rate, *i.e.*, the speed at which a link takes samples of its loads, and the sampling window, *i.e.*, the duration of the time over which the samples are kept in memory.

The discussion in Section 3.2.4 suggests that the sampling rate should be fast enough to obtain accurate parameter estimates, *i.e.*, $\Delta^{-1} \geq 2\mu_i$. Estimates are based on samples within a *moving window*[4] so the size of the window might impact the routing performance. In the following we shall vary the sampling rate and sampling window size to identify the set of operational values. The results show that the performance of the FTAR routing scheme is robust to the selection of sampling rate and sampling window size, unless very poor choices are made.

Fig.3.4 shows the performance of our routing scheme for different sampling periods and window sizes. Observe that when the sampling window is small, *i.e.*, equal to 0.01 time units, the routing performance in terms of routed volume is unsatisfactory. Indeed if the sampling window is not large enough we are not able to capture the link load dynamics. In addition, note that when the sampling rate is small, *i.e.*, with a sampling period of 1 time unit, the routing performance also deteriorates. This is consistent with our assertion in Section 3.2.4 that if the sampling is not done frequently enough, we will not have sufficient samples to be able to estimate the parameters for the Ornstein-Ulenbeck model.

Note that other than the specific cases described above, the routing performance is robust to the choice of sampling rate and sampling window size. In the sequel we will use a sampling period equal to 0.1 time units and a sampling window size equal to 1.5 time units.

[4]It is also feasible to use an "exponentially weighted-averaging" mechanism to estimate these parameters. The size of the moving window corresponds to the value of the exponential weighting factor, *i.e.*, the larger window size corresponds to the selection of the exponential weighting factor in favor of the load *history* over the *current* load.

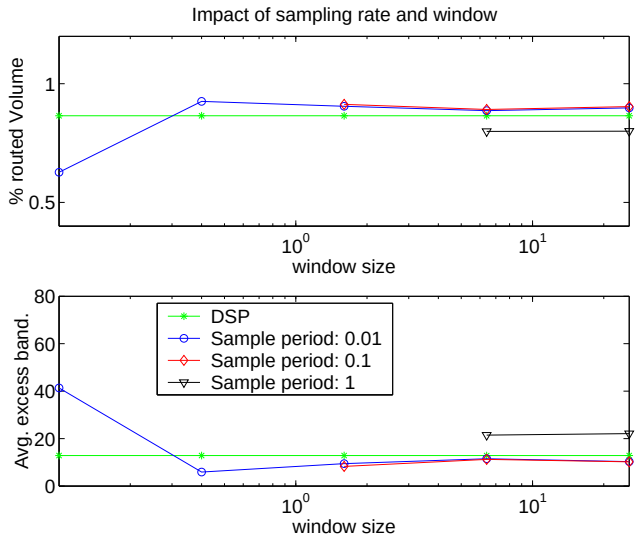

Figure 3.4: Performance as a function of link load sampling rates and sampling windows.

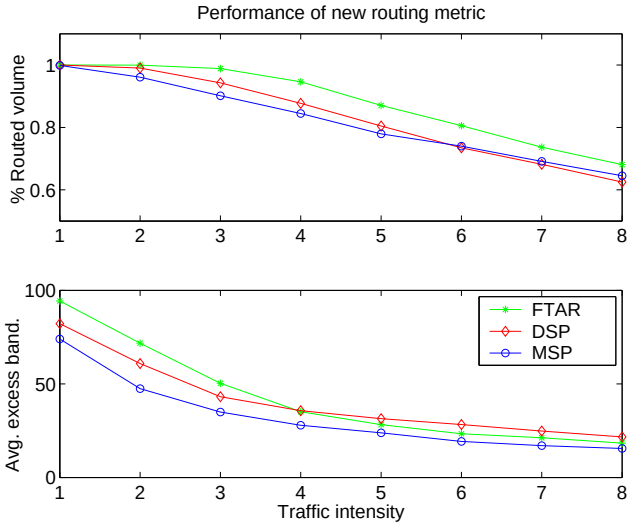

Figure 3.5: Performance gain by FTAR over DSP and MSP.

3.3.3 Performance gains using predictive flow-time-aware routing

Let us now compare the performance of FTAR with DSP and MSP. In Fig.3.5 we show a typical result for the case with exponential flow holding time distributions. We see that FTAR improves the routing performance over both DSP and MSP, by up to 10% in terms of routed volume. We observe that FTAR performs consistently better than MSP, and that only in the heavily loaded regime where FTAR is supporting a higher traffic volume, does its average flow-perceived bandwidth seen by flows become less than that for DSP. Note that in the lightly loaded regime the flow blocking performance of FTAR is better than DSP and MSP, and FTAR also provides better performance in terms of average flow-perceived bandwidth. This is surprising since the network using FTAR is admitting a higher number of flows. Hence the overall routing of traffic must be significantly improved by using a predictive flow-time aware routing mechanism.

3.3.4 Concave or additive path metrics: choice of mesh routing algorithms

To determine good path between a pair of source-destination nodes for an incoming flow, one often resorts to a notion of "shortest path" or "widest path". On the one hand, the construction of a *shortest* path often proceeds by adding up link metrics. On the other hand, the construction of a *widest* path usually involves taking the minimum (a "concave" operation) of several link metrics. It is not entirely clear what routing metrics and their associated algorithms one should use for a specific routing scenario, though [41] suggests that the inverse of the residual bandwidth might be a good additive routing metric to achieve network load-balancing. Note that in a simple parallel topology like the one we used in the previous sections the "shortest" and "widest" routing schemes are equivalent, *i.e.*, the difference arises only when there are multi-link paths in question.

In the context of predictive flow-time aware routing, we believe the choice of routing strategy, *i.e.*, "shortest" or "widest" criterion, depends on the characteristics of the incoming flow and the corresponding network load condition. In particular, we note that the

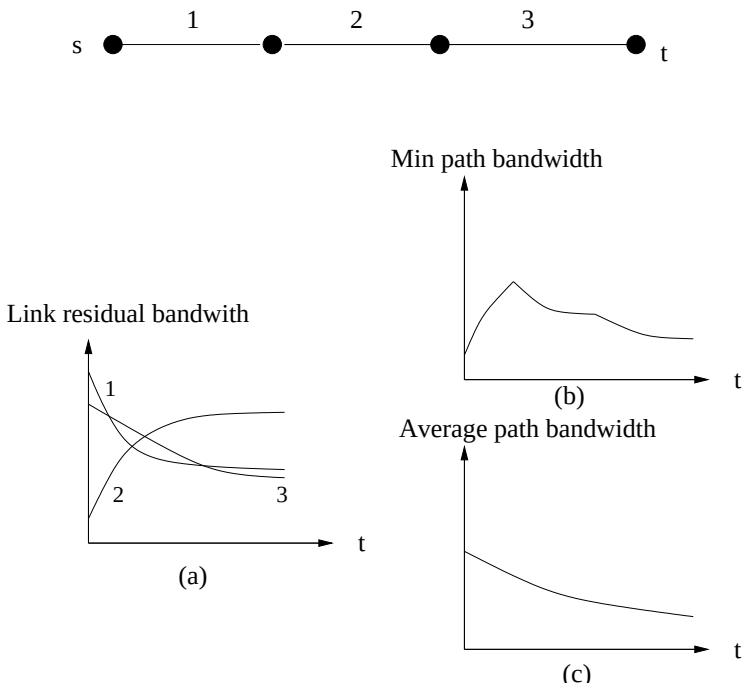

Figure 3.6: Link metrics: concave versus additive.

"dominating link" on a path, *i.e.*, the link that exhibits the "worst" load level, might vary during a flow's sojourn in the network. Fig.3.6 illustrates a path with 3 links. Fig.3.6-(a) shows the link load dynamics. Fig.3.6-(b) shows the load dynamics for the "bottleneck" link of the path, whose identity varies over time, *i.e.*, $2 \rightarrow 1 \rightarrow 3$. Fig.3.6-(c) is the load dynamics averaged over the three links. Fig.3.7 shows the performance comparison between the shortest-widest routing scheme using concave metric and the shortest path routing scheme using additive metric. We see that the routing scheme using additive metric outperforms the routing scheme using concave metric, by up to 12% in terms of blocking rate and by up to 120% in terms of average flow-perceived bandwidth.

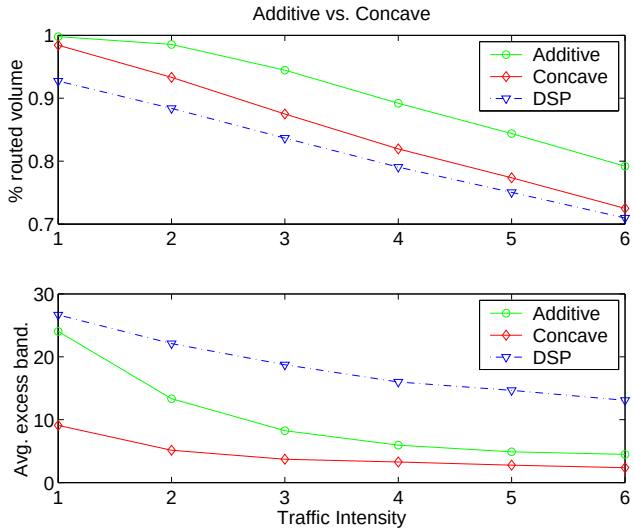

Figure 3.7: Performance comparison for concave versus additive path metric.

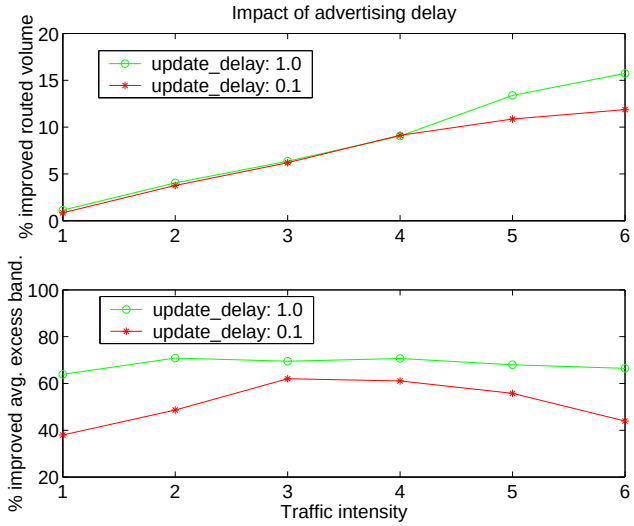

Figure 3.8: Performance improvement for FTAR over DSP for different advertising delays.

3.3.5 Effect of state advertising delays

As often is the case in practice, there are delays involved in link state broadcasts. Since dynamic routing schemes make use of link states, it is important to gage the impact these delays have on routing performance. In this section we compare the performance of FTAR and DSP as such delays increase. In particular, we will have a "slow update" scenario, where the link states are updated every 1 time unit, and a "fast update" scenario, where the link states are updated every 0.1 time units. These may correspond to networks with different geographical coverage, *i.e.*, long versus short-haul networks, or simply different limitations on the signaling overheads. We will use delayed link load in computing routing metrics for FTAR and DSP. As shown in Fig.3.8 the performance improvement by FTAR over DSP is more significant when the advertising delays are larger. This confirms our intuition in that the larger delays lead to a diminishing effect on the routing performance of the "current" link states, or alternatively, the more significant contribution by the long-term average load information, which is captured and utilized by FTAR.

3.3.6 Time-stamping mechanism

It is of interest to compare the routing performance using delayed link states, as presented in the previous section, with that where a "time-stamping" mechanism is used to determine exactly the delay associated with a given link state, *i.e.*, the links attach a time-stamp to the link states when they are advertised. As discussed earlier when making routing decisions routers can use this time-stamp information to determine the delay of link states and thus estimate the expected flow-perceived load according to (3.5). Fig.3.9 shows the performance improvement achieved by FTAR augmented with time-stamp over FTAR without the knowledge of link state advertising delays. We see that this time-stamping scheme improves the routed volume by 4% and average flow-perceived bandwidth by 25%. Moreover, we note that when the update delay is larger, the performance improvement obtained by the time-stamping mechanism is more significant. This is intuitive considering the fact that the

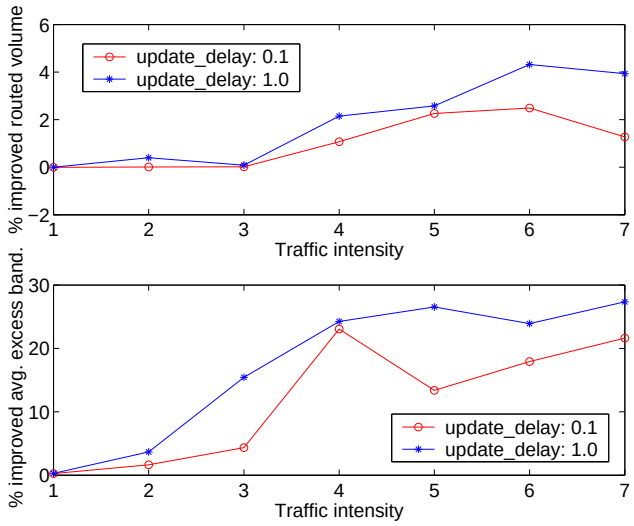

Figure 3.9: Performance improvement by time-stamping mechanism for different broadcast delays.

difference in routing metrics increases when the update delay increases between the cases with and without time-stamps, and hence the difference in the routing decisions.

3.3.7 Bursty arrivals: Markov modulated Poisson process

In the previous simulations we modeled the flow arrivals by Poisson processes. This is a relatively "smooth" random process. In this section we examine the effect of a more bursty arrival process. Specifically, we use Markov modulated Poisson process (MMPP) to model the flow arrivals. There are two "modulating" states, "high" and "low". In each state traffic flows arrive as a Poisson process. We will consider two MMPPs with different flow arrival rates in the "high" state. For the first, the flow arrival rate in the "high" state is 3 times the mean given in Table 3.1. For the second, the flow arrival rate in the "high" state is 1.5 times the mean given in Table 3.1. In the "low" state, traffic flows arrive with rate 1/3 of the mean given in Table 3.1, for both MMPPs. Besides the rates associated with the "high" and "low" states, the MMPPs are also characterized by the mean time they stay at "high" and "low" states. For the first MMPP, we set the mean time at "high" state and "low" state

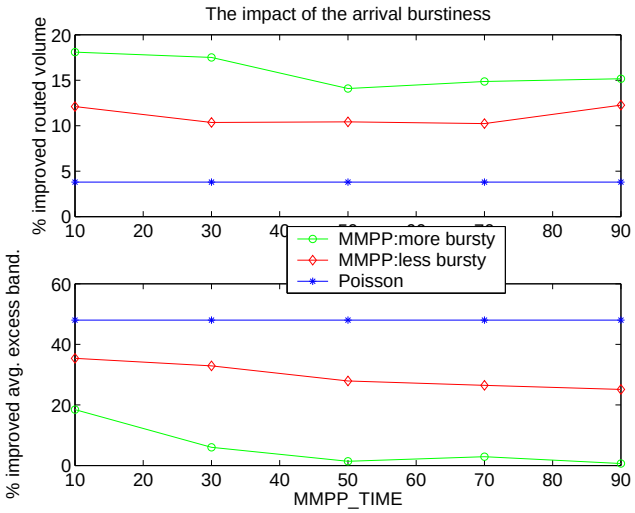

Figure 3.10: Performance improvement of FTAR over DSP with bursty flow arrivals.

to be $0.5 \cdot$ MMPP_TIME and $1.5 \cdot$ MMPP_TIME, respectively, where MMPP_TIME is a scaling variable which we vary from 10 to 90 time units. For the second MMPP, we set the mean time at both modulating states to be MMPP_TIME. The flow holding time is again exponentially distributed, with mean 1 time unit. Note that the first MMPP is more bursty than the second MMPP.

In Fig.3.10 we compare the performance improvement for FTAR over DSP, under different flow arrival processes. Observe that as the flow arrival process becomes more bursty the improvement in terms of routed volume increases, while the improvement in terms of average flow-perceived bandwidth decreases. These seemingly diverging trends make sense, since as FTAR allows increasing traffic load into the network, the average flow-perceived bandwidth reported by the (larger amount of) supported traffic decreases. This indicates that in an operating regime with bursty flow arrivals, it will be beneficial to use information on link load dynamics in addition to the "current" link load.

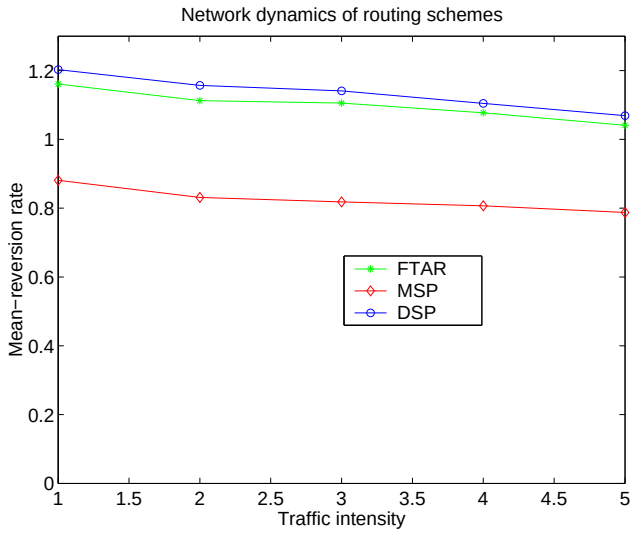

Figure 3.11: Impact of routing on link load mean-reversion rate

3.3.8 Mean reversion under various routing schemes

In the previous sections we explored three different routing algorithms, *i.e.*, DSP, FTAR and MSP. These algorithms use different link metrics when they compute the *shortest* path. We observe that the performance of a certain routing scheme depends on the network load dynamics, and the network load dynamics in turn depends on the chosen routing scheme. Note that in our parametric model for link load dynamics, the parameter α_i characterizes the rate at which the the link i's load *reverses* to its mean ρ_i. In our FTAR scheme we use a moving-window mechanism to estimate ρ_i and α_i. The estimated mean-reversion rate α_i is not used by DSP and MSP when making routing decisions. However, it is of interest to compare the average of estimated α_i over all the links in the network, when one uses different routing schemes. Let us denote this average to be α^r, where $r \in \{\text{DSP}, \text{FTAR}, \text{MSP}\}$. The higher the value of α^r, the more "responsive" the corresponding "mean-reversion" process is for the given routing scheme, *i.e.*, the stronger the "force" pulling the load process back toward its mean.

From Fig.3.11 we see that in general $\alpha^{\text{MSP}} < \alpha^{\text{FTAR}} < \alpha^{\text{DSP}}$. Thus if the routing scheme is MSP, *i.e.,* using the mean link load to guide the placement of the incoming flows, the mean-reversion is least responsive. This is intuitive since MSP is insensitive to the temporary deviation of the link loads from their means, *i.e.,* the routing decisions in this scheme contribute the least to the mean-reversion. Alternatively, if the routing scheme is DSP, then the routing decisions reinforce the mean-reversion by sending less flows to the highly-loaded links, allowing them to get back to the mean load, or directing more flows to the lightly-loaded links, allowing them to load up to the mean load. The FTAR routing scheme exhibits a degree of mean-reversion behavior in between DSP and MSP. We believe that its performance advantage derives from its ability to strike a better balance between the "natural" mean-reversion (determined by flow departures) and the routing-reinforced mean-reversion.

3.4 Application: routing max-min rate adaptive sessions

In the previous sections we showed that by using a link metric associated with the expected flow-perceived load, the routing performance improves in terms of both routed volume and average flow-perceived bandwidth. The former metric corresponds to the ability of the network to support traffic flows having minimal guaranteed bandwidth requirement. The latter metric corresponds to the *potential* for the admitted traffic flows to improve their "achieved" performance by sharing the excess bandwidth in addition to the guaranteed minimal rate. In this section we show by simulation that in a *max-min* bandwidth sharing framework the proposed routing scheme can indeed realize the potential and yield improved "achieved rate". We assume that upon arrival and departure of the flows the *excess* bandwidth allocated to the ongoing flows are instantaneously re-computed according to the max-min rate allocation scheme[45], and the traffic sources are responsive enough to adjust their transmission rates accordingly. In the following we examine the performance improvement achieved by our routing scheme in terms of the additional bandwidth seen by flows, *i.e.,* the *average*

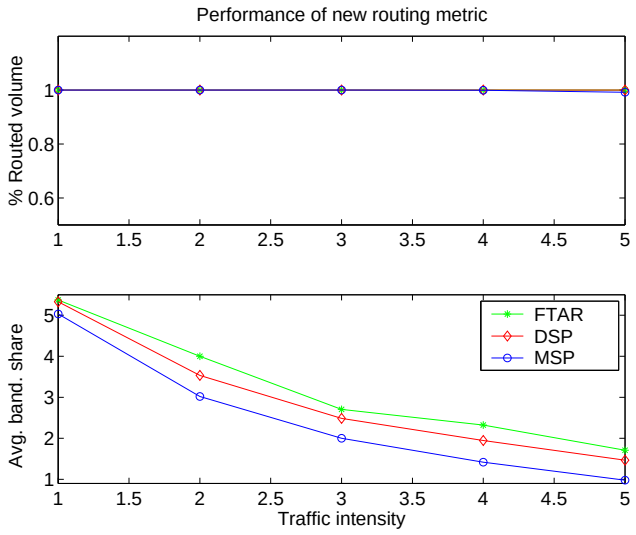

Figure 3.12: Performance comparison of FTAR, DSP and MSP with max-min fair sharing.

flow-achieved bandwidth, which is measured by first taking sampled-average of the additional bandwidth allocated to the individual flows during their sojourn in the network, then averaging over all the departed flows.

In Fig.3.12 we show a performance comparison for FTAR, DSP, and MSP routing schemes. We know from the previous simulations that the blocking performance of FTAR is superior to the baselines. To highlight the capability of FTAR in obtaining improved max-min shared rates, here we show an operating regime where the load is light, *i.e.*, no blocking occurs for all routing schemes. In Fig.3.13 we plot the % improvement of FTAR over DSP and MSP, in terms of average flow-achieved bandwidth. An improvement of 10%-20% is achieved over DSP and the improvement over MSP can be up to 70%. Similarly in Fig.3.14 we plot a performance comparison of FTAR versus DSP and MSP when there is an update delay of 0.05 unit. In this case while their performance in terms of average flow-achieved bandwidth become similar as traffic load increases, the performance of FTAR in terms of routed volume is much better than DSP and MSP.

In principle the max-min bandwidth sharing is fair in the sense that it does not dis-

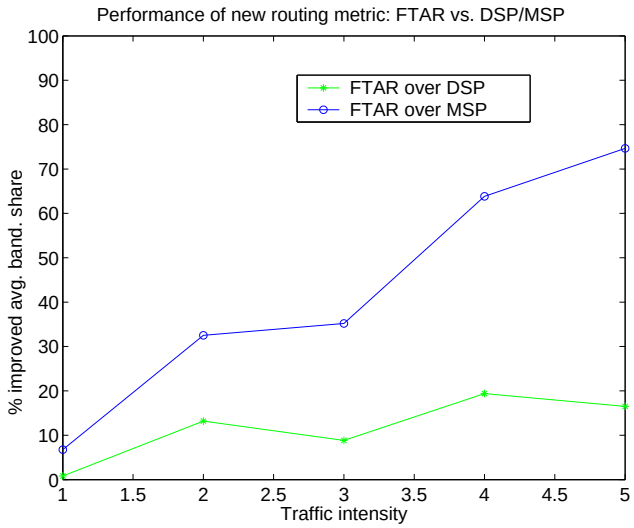

Figure 3.13: Performance improvement for FTAR over DSP and MSP with max-min fair sharing.

criminate against flows traversing long routes. Bandwidth sharing schemes used in practice, *e.g.*, proportional-fair sharing, or TCP will however do so. To study the performance of the proposed routing scheme when bandwidth sharing schemes *do* discriminate against long flows, we considered a *weighted max-min* sharing scheme where larger (smaller) weights are given to the flows that traverse shorter (longer) routes. This corresponds to larger (smaller) amount of bandwidth being allocated to the flows that traverse shorter (longer) routes. In Fig.3.15 we plot the performance results for the light-loaded regime, and Fig.3.16 shows the percentage improvement of FTAR over DSP and MSP. We observe that the performance advantage of FTAR over DSP/MSP is consistent, even with the weighted max-min sharing scheme.

3.5 Conclusion and discussion

In this chapter we proposed a new dynamic routing scheme which improves the overall performance *achieved* by stream-based flows during their sojourn in the network. The novelty

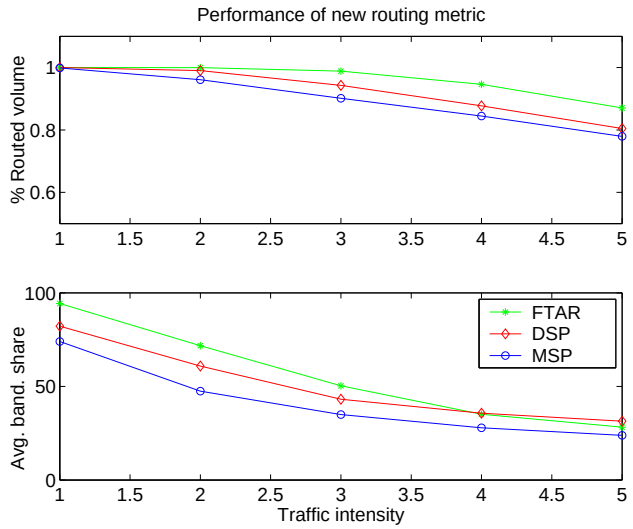

Figure 3.14: Performance comparison of FTAR, DSP and MSP with max-min fair sharing and update delay 0.05 unit.

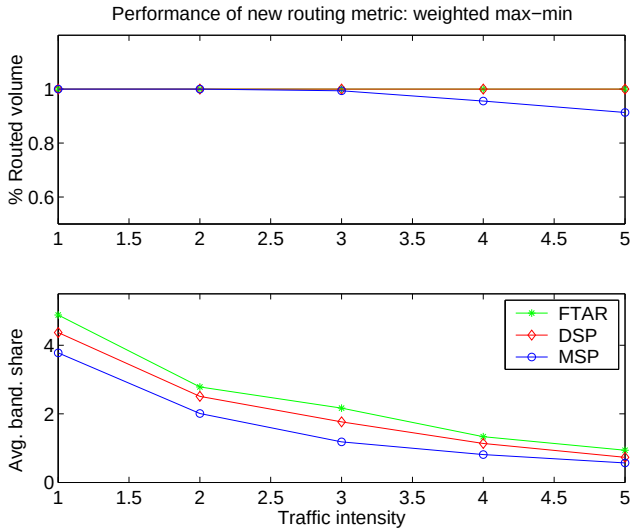

Figure 3.15: Performance comparison of FTAR, DSP and MSP with weighted max-min fair sharing.

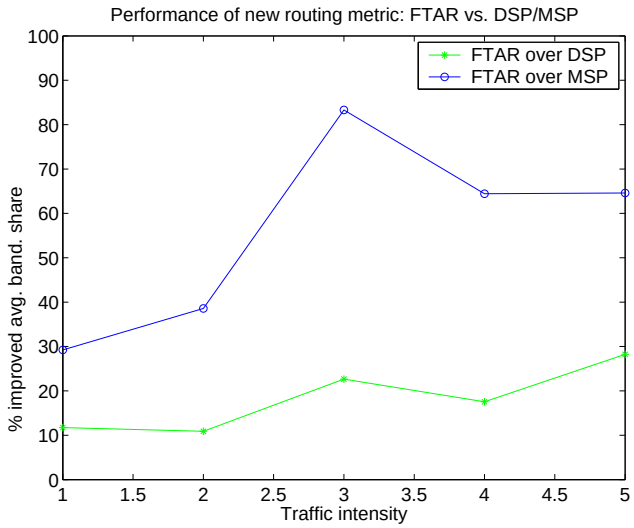

Figure 3.16: Performance improvement for FTAR over DSP and MSP with weighted max-min fair sharing.

of our approach lies in (1) the identification of the notion of *expected flow-perceived load*, which quantifies the "potential" for improvement of the user's performance, that exists at a given link from a specific flow's perspective, (2) the construction of a practical routing algorithm which realizes the above potential, based on an auto-regressive load model and the prior information on flow holding time. For a large class of traffic and service models, *e.g.*, VBR and rate adaptive applications, an effective use of our approach will result in better flow QoS. Specifically, we constructed a routing algorithm that aims at minimizing expected flow-perceived load during a flow's sojourn in the network. We showed that this routing algorithm leads to not only better load balancing in the network, but also improved flow-perceived performance. This allows the flows admitted to the network to realize a greater share of "achieved" bandwidth, in addition to their minimal requested amount.

The implementation of the proposed routing scheme would require updating routing software. We use prior information on the holding time of the traffic flows. This can be either presented by the traffic flows upon arrival to the network, or obtained through

traffic statistics gathered by the network operator. In addition, routers in the network need to maintain link load models. The effort here includes estimating the parameters of the model and advertising the estimated parameters along with current link loads. This implies additional computational and signaling overhead. However, we note that in a distributed routing environment a given router need only maintain the link load models for the adjacent links, which scales at most linearly with the number of the routers in the network[5]. Moreover, these estimated parameters are "stable" since they correspond to the mean and the rate of variation for a quasi-stationary stochastic process, thus they need not be updated as frequently as the current link load. It is shown in Section 3.3.5 that FTAR achieves higher performance gain with larger advertising delays when compared to DSP. This suggests FTAR is more robust than DSP when link state advertisements become less frequent, enabling reduced overhead by using larger advertising delays. These observations lead us to believe that the performance advantage of our routing scheme outweighs concerns with overhead. In conclusion, the routing designer can improve the routing performance of the stream-based flows by taking advantage of information regarding link load dynamics and flow holding time, without having to significantly increase the routing overheads.

3.6 Appendix: Parameter estimation

To estimate the afore-mentioned parameters, we observe that the solution for the stochastic differential equation (3.3) can be expressed as follows,

$$X_i(t) = e^{-\alpha_i t} X_i(0) + \sigma_i \int_0^t e^{-\alpha_i (t-u)} dB(u) + \alpha_i \rho_i \int_0^t e^{-\alpha_i (t-u)} du,$$

or equivalently,

$$X_i(t) = e^{-\alpha_i t} X_i(0) + e^{-\alpha_i t} I(t) + \alpha_i \rho_i e^{-\alpha_i t} \int_0^t e^{\alpha_i u} du,$$

where $I(t) = \sigma_i \cdot \int_0^t e^{\alpha_i u} dB(u)$ is an Itô integral.

[5] That is, in a fully-connected network. In a mesh network it grows much slower.

Given the observations of $X_i(t)$ at time $t_1, t_2, \cdots t_n$, *i.e.*, $x_{t_1}, x_{t_2}, \cdots x_{t_n}$, the likelihood function is given by

$$g(x_{t_1}, x_{t_2}, \cdots x_{t_n}; \alpha_i, \rho_i, \sigma_i) = \prod_{i=1}^{n} \frac{1}{\sqrt{v_i}} f(\frac{x_{t_i} - m_i}{2v_i}),$$

where $f(x)$ is the standard Gaussian density function, $m_1 = \rho_i$, $v_1 = \frac{\sigma_i^2}{2\alpha_i}$, and for $i > 1$,

$$m_i = e^{-\alpha_i(t_i - t_{i-1})} x_{t_{i-1}} + \rho_i(1 - e^{-\alpha_i(t_i - t_{i-1})})$$

and

$$v_i = \frac{\sigma_i^2}{2\alpha_i}(1 - e^{-2\alpha_i(t_i - t_{i-1})}).$$

The maximum likelihood estimators α_i, ρ_i, and σ_i take on the values that maximize

$$g(x_{t_1}, x_{t_2}, \cdots x_{t_n}; \alpha_i, \rho_i, \sigma_i).$$

Note that if $t_i - t_{i-1} = \Delta$, *i.e.*, the observations are made at the regularly spaced time instances, the above likelihood function is the same as the joint density of the observations of the following discrete-time Gaussian AR(1) process,

$$Y_n - \rho_i = e^{-\alpha_i \Delta}(Y_{n-1} - \rho_i) + Z_n,$$

if the observed values for $\{Y_n\}$ is such that $y_i = x_{t_i}$, for $i = 1, 2, \cdots n$. Here $\{Z_n\}$ is a sequence of uncorrelated random variables with zero mean and variance $\frac{\sigma_i^2(1-e^{-2\alpha_i\Delta})}{2\alpha_i}$.

To estimate the parameters $\rho_i, \beta_i = e^{-\alpha_i\Delta}$, and $\sigma_i'^2 = \frac{\sigma_i^2(1-e^{-2\alpha_i\Delta})}{2\alpha_i}$, we note that

$$\rho_i = \frac{1}{n}\sum_{i=0}^{n-1} x_i,$$

and rewrite the above AR(1) process to be

$$Y_n' = \beta_i Y_{n-1}' + Z_n.$$

Since $0 < \beta_i < 1$, $\{Y_n'\}$ is causal, we have $Y_k' = \sum_{j=0}^{\infty} \psi_j \cdot Z_{k-j}$, and $\psi(z) = \sum_{j=0}^{\infty} \psi_j z^j = \frac{1}{1-\beta_i z}$. Hence $E[Z_n \cdot Y_{n-1}'] = 0$, and $E[Z_n \cdot Y_n'] = \sigma_i'^2$. We note

1. $E[Y'_n \cdot Y'_{n-1}] = \beta_i E[Y'^2_{k-1}] + E[Z_n \cdot Y'_{k-1}]$, hence

$$\beta_i = \frac{E[Y'_n \cdot Y'_{n-1}]}{E[Y'^2_n]} = \frac{\sum_{i=2}^{n}(y_i - \rho_i)(y_{i-1} - \rho_i)}{\sum_{i=1}^{n}(y_i - \rho_i)^2}.$$

2. $E[Y'^2_n] = \beta_i E[Y'_{n-1} \cdot Y'_n] + E[Z_n \cdot Y'_n]$, hence

$$\sigma'^2_i = E[Y'^2_n] - \beta_i E[Y'_n \cdot Y'_{n-1}] = \frac{\sum_{i=1}^{n}(y_i - \rho_i)^2 - \beta_i \sum_{i=2}^{n}(y_i - \rho_i)(y_{i-1} - \rho_i)}{n}.$$

The parameters of the original link load process can be expressed as

$$\alpha_i = -\frac{\ln \beta_i}{\Delta},$$

and

$$\sigma_i^2 = \frac{2\alpha_i \sigma'^2_i}{1 - e^{-2\alpha_i \Delta}}.$$

Chapter 4

Online Distributed Failure Protection Algorithms In WDM Networks

4.1 Introduction

The popularity of the Internet has resulted in increasing demands for data traffic. This forces service providers to seriously consider new infrastructures that meet these demands. Wavelength Division Multiplexing (WDM), which allows a single fiber to carry multiple signals simultaneously, is thought to be a promising candidate to address possible bandwidth shortages on the Internet. While the enormous amount of bandwidth provided by WDM may help alleviate the mounting pressure for higher access speed, it also makes protection/restoration a very important issue in network management. For example, current technology allows up to 128 wavelengths to be multiplexed in a single fiber, each with a data rate of up to 10 Gbps. This roughly translates into millions of telephone calls on a single fiber. It is easy to see the catastrophic consequence a fiber cut may cause without appropriate protection mechanisms in place.

Different types of protection schemes have been developed for optical networks[70]. Many existing transport networks use dual SONET (synchronous optical network) rings. Dual rings are simple topologies which contain two separate paths between any pair of nodes, making them resilient to single link (or node) failures. Although simple and fast, the

direct application of ring architectures in WDM networks brings a number of problems. It is well known that ring-structured protection schemes typically rely on excessive capacity redundancy. By contrast, one can provide protection with substantially less spare capacity on mesh networks[16]. Protection schemes on mesh optical networks were intensively studied in the early 1990s[70, 16, 69, 54, 17]. Nevertheless, mesh-based SONET networks were not widely used due to certain inadequacies, notably a slow restoration process sometimes taking more than 2 seconds[54]. Note that typical Digital Cross-Connect Systems (DCS) in transport networks have very limited functionality. Hence only simple restoration algorithms were developed for mesh networks in the aforementioned literature.

Recently emerging Optical Cross Connect (OXC) on WDM networks are redefining survivability issues of mesh networks. Unlike their predecessor DCS, intelligent OXCs function much more like ATM switches or IP routers. They offer dynamic configuration via light path switching and allow many management tasks to be carried out in a distributed manner. Because of the preponderance of IP traffic, IP-oriented control planes are being considered for WDM-based optical networks in order to provide seamless data transport [26, 4, 21]. The goal is to provide integrated functionality such as light path routing, signaling, and restoration. This brings forth a significant shift in the management paradigm from centralized control to distributed control[13].

This shift in management paradigm has a significant impact on the design of protection solutions for WDM networks. The two major issues of network survivability—restoration time and resource efficiency, can now be addressed effectively. Many protection/restoration schemes have been proposed to achieve improved performance. In general, these schemes can be categorized in terms of their computation time as real time versus pre-provisioned, or their traffic rerouting scheme as link-based versus route-based, or their route computation mechanism as centralized versus distributed[12].

The real time approach[27, 32] computes the restoration path *after* a failure event, trying to activate the restoration process in the most resource efficient manner. Since it

needs time to establish the alternate restoration route the real time approach is likely to be slow, and moreover, there is no guarantee one such path exists when a failure occurs. By contrast, the pre-provisioned approach[34, 2] computes the protection paths before a failure occurs, and based on the result of the route computation establishes necessary states at the relevant OXCs in preparation for the switch-over actions upon the detection of a failure event. Evidently, the pre-provisioned approach allows higher restoration speed, though it might be less resource-efficient.

From the perspective of the mechanisms used to compute routes, current work regarding the optimization of the resource utilization for failure protection and restoration can be categorized into centralized [16, 31, 56, 50], or distributed approaches[12, 30]. For a centralized restoration algorithm, the restoration paths for all demands are computed at a central controller assuming up-to-date network state information is available. After their computation at the central controller the restoration routes are distributed to the OXCs to establish (restoration) routing tables. The centralized computation can be very resource-efficient, if the central controller possesses up-to-date information regarding network topology, link capacity and traffic demands. However, maintaining up-to-date information on topology and link capacity require frequent signaling communication between a central controller and nodal databases, and thus can be costly. The centralized computation may not be able to scale with a large number of demands. Moreover, the "batch" computation assuming knowing all traffic demands may work well in conventional telecommunication networks where the traffic demand is quasi-static, but it is not suitable in a dynamic, data-centric environment such as the bandwidth-on-demand paradigm now considered by Optical Domain Service Interconnect (ODSI) [15] and Internet Engineering Task Force (IETF) [13]. With batch computation, any incremental change in the traffic demand will cause existing paths to be re-computed and some of these paths reconfigured, which is not desirable. By contrast, distributed computations may not be as resource-efficient as their centralized counterparts, but they are scalable, easier to maintain, and do not have a single point of failure.

The current solutions on restoration path computation can also be classified into either path-based[34, 61] or link-based approaches[27, 71]. In the former case, upon detection of a failure event by the destination node of the connection, a notification is sent to the traffic source where a backup path is activated. In the latter case a failure event is detected and dealt with locally, *i.e.*, a "detour" is set up around the failed link/node. On the one hand, the path-based approach works well in seeking out an end-to-end resource-efficient backup path, and this backup path can be made failure independent, *i.e.*, link-disjoint (or node-disjoint) from the primary path, thus all the potential failures on the primary path can be protected by this backup path. However, to activate this backup path incurs end-to-end signaling and thus a longer response time. On the other hand, link-based approach may not be able to establish the optimal protection path from an end-to-end perspective, and it may require "failure isolation", *i.e.*, the identification of the failed link/node, but the speed at which a backup is set up is much higher.

In this chapter, we propose a novel link metric and several distributed routing algorithms that maximize wavelength "sharing" among independent protection paths. We formulate the problem in the link-based restoration context, then extend the proposed scheme to a generic node-based approach. The goal is to devise a generic algorithm that efficiently exploits the potential sharing opportunities among the protection paths assuming no concurrent protection paths need to be activated. These algorithms support on-demand path computation, so information about the complete traffic demands is not required. In practice, the proposed algorithms can be easily implemented as an extension of the existing IP routing protocols, e.g., open shortest path first (OSPF). The protection paths are optimized to reduce the wavelength redundancy while working/primary paths are assumed to be routed using minimum-hop paths. This separate optimization of the working and protection path is a design choice we made. It is partly motivated by [50], where it was shown that joint working/protection path computation significantly complicates the path computations with only marginal gain in the performance. We leave to future study the investigation of the

impact of this design choice.

In Section 4.2 we formulate two integer programs which capture the idea of wavelength sharing in protection path computation. Several observations are made to motivate the design of our distributed algorithms. In Section 4.3 we introduce our "bucket-based" link metrics and the corresponding routing algorithms. A modification to the link-based restoration mechanism, termed "node"-based restoration, is also discussed. In Section 4.4 we present simulation results and evaluate the performance of the proposed solution. We conclude the chapter with future directions in Section 4.5.

4.2 Problem setup

Let us consider a WDM network $G(N, E)$, where N is the set of nodes and E is the set of links. Suppose there exists a set of demands U which request light-paths to be established across the network. These demands are protected by link-based restoration, *i.e.,* each link (i, j) on the working path of demand $u \in U$ is protected by an alternative path connecting i and j. Let $x^u_{(i,j)}$ denote the number of wavelengths reserved on link (i, j) in order to carry the traffic of demand u. Similarly let $y^{(i,j,u)}_{(m,n)}$ be the number of wavelengths reserved on link (m, n) for demand u in case link (i, j) fails. Hence $x^u_{(i,j)}$ and $y^{(i,j,u)}_{(m,n)}$ denote the routing of working and protection paths respectively. We make the following assumptions:

- The traffic demands are in the form of unit wavelength requests. *i.e.,* $x^u_{(i,j)} \in \{0, 1\}$ and $y^{(i,j,u)}_{(m,n)} \in \{0, 1\}$.
- The working path of u (*i.e.,* $x^u_{(i,j)}$) is determined by the minimum-hop path.
- The number of wavelengths on each link is unconstrained.
- Only single link failure may occur at any instance of time.

The objective of a protection routing algorithm is to determine the protection paths for every u such that the total number of wavelengths reserved for protection is minimized.

This can be formulated as an optimization problem:

$$\min\{\sum_{(m,n)\in L} w_{(m,n)}\}$$

subject to:

$$\sum_n y_{(m,n)}^{(i,j,u)} - \sum_n y_{(n,m)}^{(i,j,u)} = \begin{cases} x_{(i,j)}^u & m = i \\ -x_{(i,j)}^u & m = j \\ 0 & m \neq i, m \neq j \end{cases}, \tag{4.1}$$

$$w_{(m,n)}^{(i,j)} = \sum_u y_{(m,n)}^{(i,j,u)}, \tag{4.2}$$

$$w_{(m,n)} \geq w_{(m,n)}^{(i,j)}, \tag{4.3}$$

$$y_{(m,n)}^{(i,j,u)} \in \{0,1\}, \tag{4.4}$$

$$y_{(i,j)}^{(i,j,u)} = 0. \tag{4.5}$$

The constraint (4.1) is related to flow conservation on the protection path. In (4.2), $w_{(m,n)}^{(i,j)}$ is the amount of traffic on link (i,j) that will be moved to link (m,n) if link (i,j) fails. To take into account the reservation sharing among non-concurrent failures and to ensure enough wavelengths are reserved for any link failure, $w_{(m,n)}$ needs to be reserved on link (m,n) , *i.e.,* (4.3) must be satisfied.

This problem formulation fits well into a centralized management paradigm where the Network Management System (NMS) may optimally configure every protection path, using the complete knowledge of the demand set U. However, such an off-line algorithm is not desirable in an environment where the demands for light-paths arrive and depart dynamically. After all, it is costly to re-configure the whole network whenever traffic demands change. Instead, an online protection routing algorithm is preferred in a dynamic environment.

An online algorithm determines the protection routing based on the *existing* network status. We do not assume that all future demands are known or the existing demands can be rerouted. Thus the objective of an online algorithm is to minimize the *marginal* wavelength

requirements for each new demand u^*. Suppose the working path $x^{u^*}_{(i,j)}$ has been determined by the minimum-hop path. Another optimization problem can be formulated to determine the protection path, *i.e.*, $y^{(i,j,u^*)}_{(m,n)}$:

$$\min\{\sum_{(m,n)\in L} \Delta w^{(i,j)}_{(m,n)}\}$$

subject to:

$$\sum_n y^{(i,j,u^*)}_{(m,n)} - \sum_n y^{(i,j,u^*)}_{(n,m)} = \begin{cases} x^{u^*}_{(i,j)} & m = i \\ -x^{u^*}_{(i,j)} & m = j \\ 0 & m \neq i, m \neq j \end{cases}, \tag{4.6}$$

$$b^{(i,j)}_{(m,n)} = \begin{cases} 0 & \text{if } w^{(i,j)}_{(m,n)} + 1 \leq w_{(m,n)} \\ 1 & \text{otherwise} \end{cases}, \tag{4.7}$$

$$w^{(i,j)}_{(m,n)} = \sum_u y^{(i,j,u^*)}_{(m,n)}, \tag{4.8}$$

$$w_{(m,n)} \geq w^{(i,j)}_{(m,n)}, \tag{4.9}$$

$$\Delta w^{(i,j)}_{(m,n)} \geq b^{(i,j)}_{(m,n)} \times y^{(i,j,u^*)}_{(m,n)}, \tag{4.10}$$

$$y^{(i,j,u^*)}_{(m,n)} \in \{0, 1\}, \tag{4.11}$$

$$y^{(i,j,u)}_{(i,j)} = 0. \tag{4.12}$$

In (4.7), $b^{(i,j)}_{(m,n)}$ is the *additional* wavelength requirement on link (m, n) if it is used by u^* to set up a protection detour for a failure of link (i, j). It is determined based on sharing reservations with other failures. For example, if $w^{(i,j)}_{(m,n)} + 1 \leq w_{(m,n)}$, no additional wavelength reservation for the protection of link failure (i, j) is necessary since a sufficient amount of wavelengths $w_{(m,n)}$ has already been reserved on link (m, n). If $w^{(i,j)}_{(m,n)} = w_{(m,n)}$, one additional wavelength needs to be reserved if the protection path for link (i, j) is to traverse link (m, n).

From this problem formulation, we note that (1) determining $y^{(i,j,u^*)}_{(m,n)}$ is equivalent to finding the *minimum-cost* path from i to j in the network $G(N, E - \{(i, j)\})$, and (2) the existing network status can be aggregated into $w^{(i,j)}_{(m,n)}$ and $w_{(m,n)}$. As a result, the

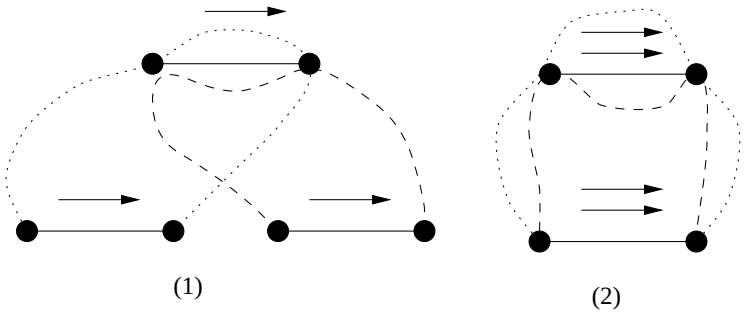

Figure 4.1: (1) Sharing; (2) No-sharing.

protection routing problem in WDM networks can draw upon the conventional *shortest path* routing algorithms in data network. In the following we propose a novel link metric that provides the necessary network state information in an aggregated form and develop an online protection routing algorithm which fits into the current Internet routing framework.

4.3 Bucket-based link metrics to maximize the sharing on protection paths

As motivated in the previous section, the problem we are faced with is to devise a distributed online restoration routing algorithm that maximizes the sharing of resources among protection paths. The key contribution we offer, lies in providing a protection scheme that is (1) bandwidth efficient; (2) computationally simple and conforming to the existing Internet routing framework; and (3) amenable to fast restoration. We propose a link-based rather than path-based approach to enable speedy restoration. We use a shortest-path algorithm which is simple and can be easily adapted from the current routing algorithms widely in use on the Internet, *i.e.*, Bellman-Ford and Dijkstra algorithms. As the center piece of the proposal, we design a unique link metric which results in a resource efficient protection mechanism.

The key idea of the proposal stems from the observation that the protection paths

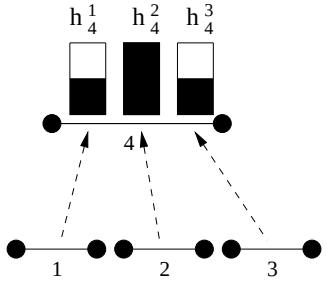

Figure 4.2: Link metrics: buckets

for different link failures can share the protection wavelengths since these paths need not to be activated at the same time, as illustrated in Fig. 4.1. This suggests an efficient "bucket-based" link state representation. In the network $G(N, E)$, each link $l \in E$ maintains a set of "buckets", $h_l = (h_l^k, k \in E, k \neq l)$, as illustrated in Fig. 4.2. Each bucket h_l^k corresponds to a failure on link k, and the "height" of the bucket, *i.e.*, the value of h_l^k, indicates the protection wavelengths that are reserved on link l for the failure event k. In terms of the notation introduced in the previous section, we have the correspondence $h_l^k = w_{(m,n)}^{(i,j)}$ for link $l = (m, n)$ and failure $k = (i, j)$. The wavelengths that need to be reserved at link l equal to the maximum of the bucket heights, *i.e.*, $\max_k h_l^k$. Thus by maintaining a sequence of buckets indexed by the failure events, we capture the necessary information on the sharing potential offered by each link. Notice that this sharing potential is a function of the failure event. For example, in Fig. 4.2, Link 4 maintains 3 buckets. Bucket h_4^2, corresponding to the failure of Link 2, is the highest. This indicates that in order to protect an additional wavelength on Link 2, Link 4 has to reserve an extra wavelength if it is selected as part of the protection route. By contrast, to protect an additional wavelength on Link 2 or 3, no extra wavelengths need to be reserved.

Equipped with the proposed link metric representation, we now describe the computational procedure for determining protection paths. It is a variant of the "shortest-widest" algorithm. The pseudo-code is exhibited in Fig. 4.3. We define the "width" $\llcorner width(l; k^*)$ of a link l with respect to a failure event k^*, as the normalized difference between the max-

imum bucket height, $\max_k h_l^k$ and the bucket corresponding to link failure k^*, $h_l^{k^*}$, *i.e.*, $l\_width(l;k^*) = 1 - \frac{h_l^{k^*}}{\max_k h_l^k}$ if $\max_k h_l^k > 0$, and $l\_width(l;k^*) = 0$ otherwise. Observe that the $l\_width(l,k^*)$ is between 0 and 1, and this value indicates the sharing capability that link l has to offer for the protection of the failure k^*.

We use a modified Bellman-Ford algorithm[73] to identify the widest paths between the end nodes of the protected link, *i.e.*, the paths that offer the most sharing. Here the width of the path p with respect to a link failure k^*, $p\_width(p,k^*)$, is defined to be the minimum of its link components, *i.e.*, $p\_width(p;k^*) := \min_{l\in p} l\_width(l;k^*)$. Observe that by this definition we implicitly establish that the marginal cost of traversing a path is dictated by that of the "narrowest" links along the path.

In the event that there are more than one such widest paths, and their widths are all 0 (*i.e.*, these paths all go through at least one link with non-zero marginal wavelength consumption), we select the one that traverses the least number of "exhausted" links, *i.e.*, the links of width 0. In all other cases of tie breaking with positive path width, *i.e.*, the marginal costs are zero, we randomly select one with widest path. Notice that this is obviously a locally optimal scheme. In other words, given only the current demand and without any knowledge about the future arrivals, the protection path we come up with is the most efficient in utilizing the sharing opportunities. See Fig. 4.3 for the shortest-widest algorithm we use to compute the protection paths.

4.3.1 Highlights of the proposal

In Section 4.4 we will evaluate the effectiveness of our proposal via simulation. Let us summarize its desirable features at this point:

- This procedure solves the optimal on-demand protection problem we formulated in Section 4.2. In the sequel, its performance improvement over a baseline scheme is demonstrated in terms of the reduction in wavelength redundancy.

- This proposal conforms to the existing Internet routing framework. In particular, it

function routing-protection($G(N, E), s, t, \mathbf{w}(\mathbf{E})$)

inputs:

$G = (N, E)$: a network;

N: the set of nodes of G;

E: the set of links of G, also the set of failure events;

s: source node in N;

t: destination node in N;

$\mathbf{h_l}$: a vector link metric associated with link $l \in E$,

$\mathbf{h}_l = (h_l^1, h_l^2, \cdots h_l^k, \cdots h_l^{|E|}, k \in E, k \neq l)$;

h_l^k: the wavelengths reserved on link l, for the protection of failure k;

returns:

working path $r(s, t)$ and associated protection paths $p[l]$ for all links $l \in r(s, t)$.

1. $r(s, t) = compute\_working\_path(s, t)$.
2. **for** each link $k^* \in r(s, t)$
3. $s\_node = source(k^*)$, $d\_node = end(k^*)$.
4. s_node performs the following:
5. **for** each link $l \in E$
6. $l\_width(l; k^*) = 1 - \frac{h_l^{k^*}}{\max_k h_l^k}$
7. **endfor**
8. $l\_width(k^*; k^*) = -1$; // "remove" link k^*
9. $(widest\_paths, widest\_width)$
10. $= compute\_widest\_paths(s\_node, d\_node)$;
11. **if** $||widest\_paths|| > 1 \quad and \quad widest\_width == 0$ **then**
12. **for** each $p \in widest\_paths$
13. $c[p] = count\_saturated(p)$;
14. **endfor**
15. $p[k^*] = \min_p c[p]$;
16. **else if** $||widest\_paths|| > 1 \quad and \quad widest\_width > 0$ **then**
17. $p[k^*] = random\_select(widest\_paths)$;
18. **else**
19. $p[k^*] = widest\_paths$;
20. **endif**
21. **endfor**

Figure 4.3: Routing-protection: a shortest-widest algorithm

is easy to modify Dijkstra or Bellman-Ford algorithms to implement our shortest-widest algorithm. The amount of link states ($O(|E|^2)$) is proportional to the product of the number of links and the number of failures [1], not the number of light paths, and hence is manageable in practice. In particular, one can implement our bucket link metric via "opaque LSA (Link State Advertisement)", an OSPF option[7]. Moreover, we note that the link states are set up in a way that is flexible enough to accommodate failure types other than link failure, *e.g.*, node failures.

- One feature of this proposal is the association between the link state/cost and the failure event. For the protection routing of different failure events, one is presented with different network topology and link states. Similarly, as network state evolves with connections arriving to (and departing from) the network, one might have to establish different protection paths for the protection of the same link failure. Our proposal exploits this feature to locate the protection path that incurs least marginal cost for a particular link failure at a particular instance of time. It is easy to extend our scheme to encompass further considerations, *e.g.*, excluding paths that traverse too many hops and thus take too much time to activate when a failure occurs.

4.3.2 A generalization: node-based protection

In the previous section we introduced a "bucket"-based link metric and a corresponding shortest-widest routing algorithm. The study is set in the context of the link-based failure detection, *i.e.*, we assume the loss of the optical signal is caused by a link failure. The implication, in relation to our routing/protection design, is that we construct the protection paths that are constrained to go from one end of the protected link to the other end, as illustrated in Fig. 4.5-1.

[1] The number of failures is equal to number of links in a link-based protection scheme. As a comparison, for a path-based protection schemes based on the notion of buckets, where working paths between a source and a destination are fixed, *e.g.*, minimum-hop paths. The number of protection link states that needs to be maintained is $O(|E||N|^2)$.

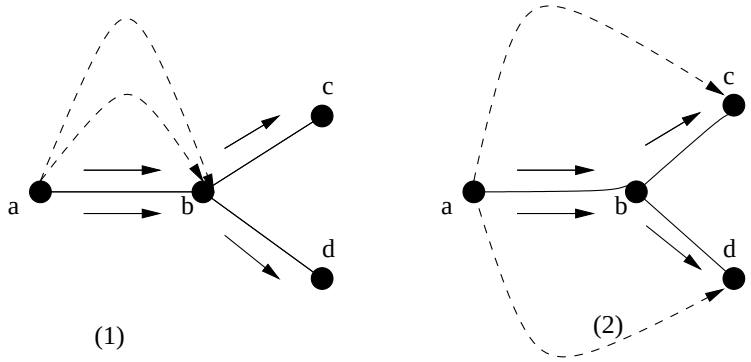

Figure 4.4: (1) Link-based computation; (2) Node-based computation.

We observe that this design possesses some obvious merits, especially the locality of the restoration operation. However, it is also worth noting that restoration efficiency is negatively impacted by this choice. By constructing detours from one end of the failure link to the other end, even when traversing demands are destined to different nodes, the group of restoration paths may unwittingly clog the "local area" and under-utilize the potential sharing capability in the network. Fig. 4.4 illustrates the problem: the working path of 2 demands share link (a, b), but diverge to node c and node d respectively. With link-based restoration, the protection paths for both demands have to go from node a to node b. This may not be desirable. Alternatively, if we set up these protection paths such that they start from node a, but end at node c and node d, there is a better chance to exploit "network-wide" sharing potential. Fig. 4.5-(2) illustrates a different restoration mechanism, where the protection path for a given link ends at the node two hops away on the corresponding working path. We call this "node"-based restoration. For example, for the protection of link (a, b), instead of going from a to b, we construct a protection path from a to c. A special case is link (c, d), for which the node-based protection path share the same end nodes as their link-based counterparts, since d is the destination and there are no nodes further down the working path.

From a practical standpoint, we note that the optical cross-connects, depicted in

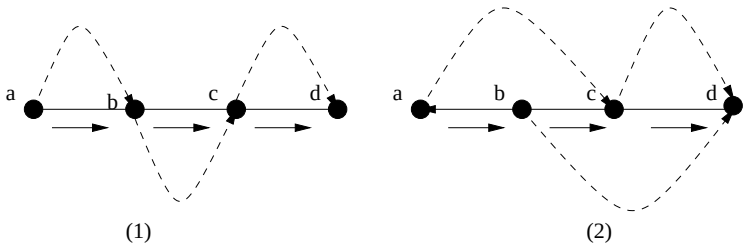

Figure 4.5: (1) Link-based computation; (2) Node-based computation.

Fig. 4.5 as the black nodes, may also cause disrupted service when they fail, in which case all the links adjacent to the failed node will "fail" simultaneously. Under this condition, the construction of the protection paths for a particular link should consciously exclude the links that are experiencing the problem at the same time. Differentiating link or nodal failures takes time and causes undesirable delays in service recovery. In certain cases it is more advantageous to be conservative, *i.e.,* to use the nodal failure model as the general model of the failure events, and treat the single link failure as a special case. The "jump-ahead" operation proposed above is well suited for provisioning protection wavelengths in this context. Fig. 4.6 contains the pseudo-code for the node-based protection path computation. The main differences from the link-based algorithm are: (1) To enable the node-based operations we need to maintain a function $next\_link(p, i)$ at node $source(i)$ which outputs the link that is right next to link i on the path p, if such a link exists. (2) As highlighted in Fig. 4.6, the major modification to the algorithm in Fig. 4.3 lies in lines 4 and 5, where we "jump ahead" to identify the end node of the protection path. Unlike the scheme in Fig. 4.3, where the protection path is constrained to go from one end of the protected link to the other, in Fig. 4.6 we start the protection path at one end of the protected path, but end it at the nodes two hops away, if such a node exists on the working path.

function node-routing-protection($G(N,E), s, t, \mathbf{w}(\mathbf{E})$)

inputs:

$G = (N, E)$: a network;

N: the set of nodes of G;

E: the set of links of G, also the set of failure events;

s: source node in N;

t: destination node in N;

$\mathbf{h}_l$: a vector link metric associated with link $l \in E$,

$\mathbf{h}_e = (h_e^1, h_e^2, \cdots h_e^i, \cdots h_l^{|E|}, i \in E, i \neq e)$;

h_i^k: the wavelengths reserved on link i, for the protection of failure k;

returns:

working path $r(s,t)$ and associated protection paths $p[l]$ for all links $l \in r(s,t)$.

1. $r(s,t) = compute\_working\_path(s,t)$.
2. **for** each link $l \in r(s,t)$
3. $s\_node = source(l), \quad d\_node = end(l)$.
4. **if** $d\_node \neq t$ **then**
5. $d\_node = end(next\_link(r(s,t), l)$.
6. **endif**
7. s_node performs the following:
8. **for** each link $l' \in E$
9. $l\_width(l') = 1 - \frac{h_{l'}^l}{\max_i h_i^l}$
10. **endfor**
11. $l\_width(l) = -1$;
12. **if** $d\_node \neq t$ **then**
13. $l\_width(next\_link(r(s,t), l) = -1$;
14. **endif**
15. $(widest\_paths, widest\_width) = compute\_widest\_paths(s\_node, d\_node)$;
16. **if** $||widest\_paths|| > 1 \quad and \quad widest\_width == 0$ **then**
17. **for** each $p \in widest\_paths$
18. $c[p] = count\_saturated(p)$;
19. **endfor**
20. $p[l] = \min_p c[p]$;
21. **else if** $||widest\_paths|| > 1 \quad and \quad widest\_width > 0$ **then**
22. $p[l] = random\_select(widest\_paths)$;
23. **else**
24. $p[l] = widest\_paths$;
25. **endif**
26. **endfor**

Figure 4.6: Node-routing-protection: a modified routing-protection algorithm

4.4 Performance evaluation

In order to evaluate the performance of our proposal for protection path routing, we performed a number of simulations. In the following we present our results associated with the network topology shown in Fig. 4.7 [38] and Fig. 2.8. Unless specifically indicated, the source and the destination node of the traffic demands are distributed uniformly across nodes in the network. The demands arrive in sequence and are routed one at a time. Each demand requests one wavelength. We measure the "redundancy", *i.e.,* the ratio between the number of protection wavelengths and that of the working wavelengths, and plot it against an increasing number of demands in the network (on the log-scaled x-axis). As a reference, we solved the integer program formulated in Section 4.2. We are able to obtain the solution for 100 demands. For 15-node network, the redundancy is 61%. We are not able to solve for larger number of demands due to the inability of CPLEX[5] and bonsaiG[29] to deal with larger-sized problems. Note that if further incoming demands are merely repetitions of the first 100 demands, then the solution for the integer program will stay the same, *i.e.,* 61%. As it stands we think it is sensible to extrapolate this value as an approximation for the ideal off-line performance with larger numbers of demands. As a reminder, we note that this is a solution that assumes complete knowledge of all the demands so it only serves as a lower bound for the performance our distributed online algorithm might achieve. We compare our proposal and a baseline online algorithm, which protects each link along the working path by a detour of minimum hop count. We will see in the sequel that as we shift the problem domain into the online context, the baseline protection scheme exhibits significant increases in redundancy. Our proposal is a worthwhile effort to push the redundancy value toward that of the ideal off-line solution.

4.4.1 Performance improvement over the baseline

We first evaluate the performance of our proposal against the baseline algorithm. The baseline algorithm operates without using the aggregate bucket information as our scheme does,

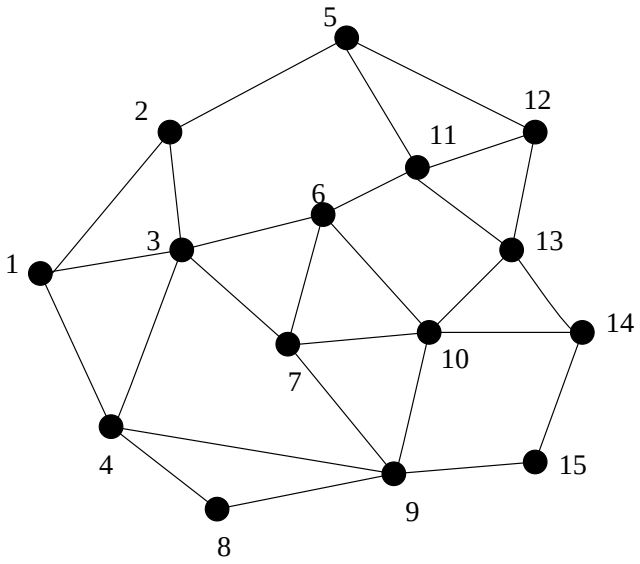

Figure 4.7: 15-node network

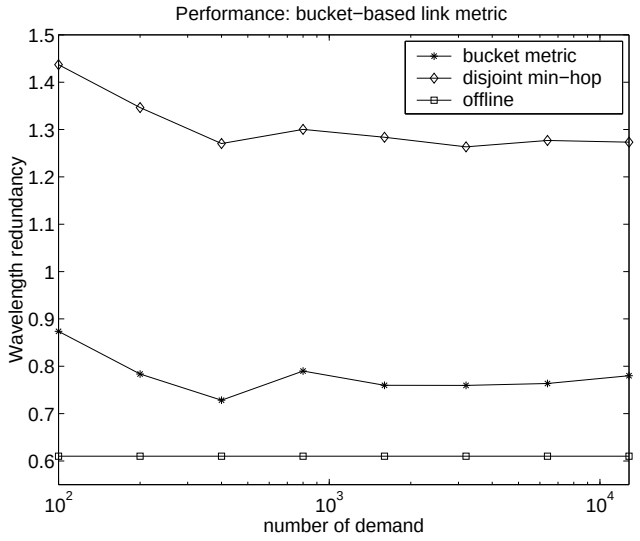

Figure 4.8: Performance evaluation: 15-node net

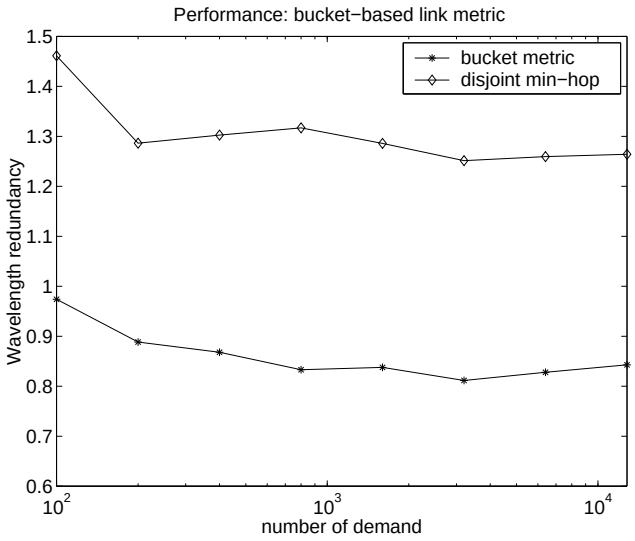

Figure 4.9: Performance evaluation: NSFnet

hence provides a legitimate reference to (1) demonstrate the dramatic increase in wavelength redundancy when we shift the problem domain from off-line computation to online computation, and (2) evaluate the impact that the additional bucket state information has on the overall wavelength utilization. We observe from Fig. 4.8 that a significant reduction in redundancy is obtained by introducing the bucket link states. In addition, we see a near-constant capability of our approach at realizing the network's sharing potential, when it is sufficiently loaded, *i.e.,* around 75% protection wavelength redundancy for a 15-node network.

4.4.2 Impact on performance of the demand structure

The demands in the previous section were drawn randomly from the (source, destination) space. It is interesting to study the change in the performance if demands exhibit locality. In particular, we are motivated to investigate the effect of the different demand locales, *i.e.,* whether they are "remote" or "co-located". In this section we respectively arrange for all the

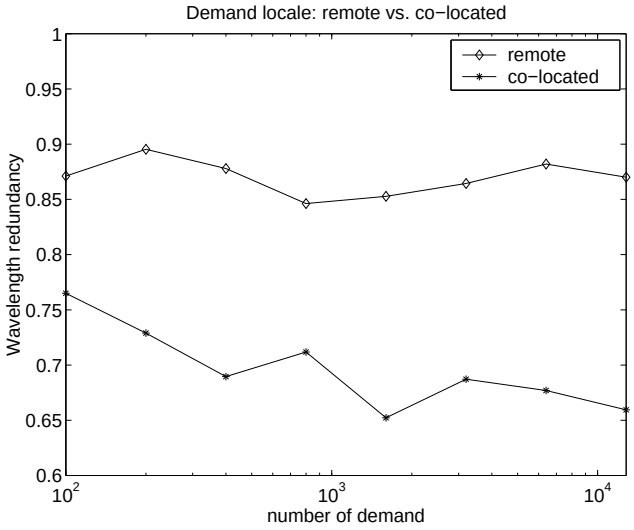

Figure 4.10: Remote vs. co-located: 15-node net

demand sources and destinations to be (1) *at least* 3 hops apart, or (2) *at most* 2 hops apart. Fig. 4.10 and 4.11 indicate that when the source and the destination are relatively remote from each other, the wavelength redundancy, *i.e.,* the ratio of the wavelengths used to protect and those used to carry primary traffic, is higher. This suggests that there exists larger potential for wavelength sharing when the traffic are more co-located. This makes sense since for the links that are close-by, their protection paths are more likely to go through same set of links, thus sharing a common set of wavelengths.

4.4.3 Using iterations to improve the solution quality

An interesting indicator of the quality of our proposed routing approach, is the degree of sharing potential that has *not* been utilized after we configure the protection paths. By allowing iteration, or re-routing of the protection paths, we can measure the degree of unrealized sharing opportunities. The “iteration” here simply means we allow the re-computation of the restoration paths with renewed aggregate network information, *i.e.,* bucket link states, reflecting the results of the previous (protection) routing computation. It is possible to find

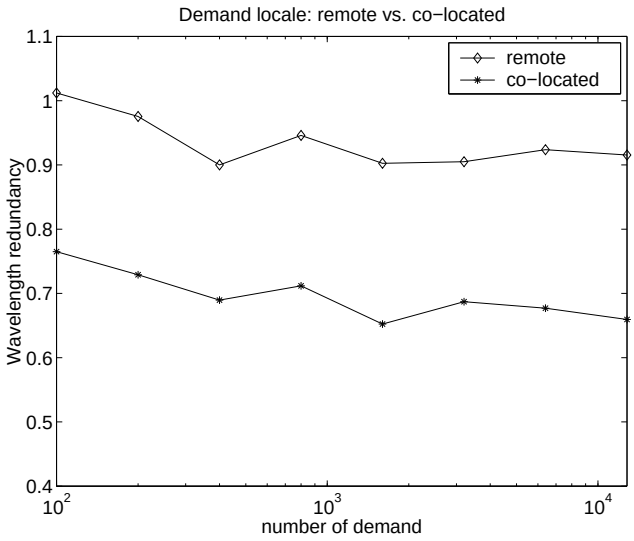

Figure 4.11: Remote vs. co-located: NSFnet

a better protection path for a particular demand by running iterations since in so doing we approach the offline solutions with complete knowledge of what other protection demands were routed. The degree of the improvement indicates how close to (local) optimality the initial configuration is. The smaller the improvement, the higher the quality of the initial configuration.

demands #	working wavelengths	ite-0	ite-1	ite-2	ite-3
100	222	202	194	194	194
1000	2174	1687	1665	1665	1665
12800	28220	21588	21531	21531	21531

Table 4.1: Iteration: 15-node network

demands #	working wavelengths	ite-0	ite-1	ite-2	ite-3
100	232	242	226	226	226
1000	2313	1969	1951	1951	1951
12800	29995	24867	24829	24829	24829

Table 4.2: Iteration: NSFnet

Tables 4.1 and 4.2 demonstrate that we obtain a marginal amount of reduction in

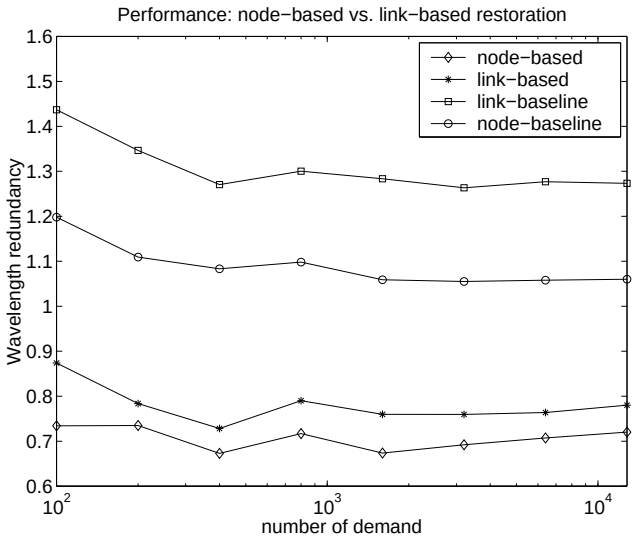

Figure 4.12: Link-based vs. node-based, 15-node net

wavelength redundancy with one iteration, and the iterations after that do not improve the performance. For example, in the case of 1000 demands, the first iteration reduces the protection wavelength consumption by 22 units, a mere 1.3% decrease. Moreover, further iterations make no additional improvement. Thus it is justified to state that our routing scheme is able to realize almost all of the sharing potential during its first run.

4.4.4 Performance evaluation: node-based vs. link-based algorithms

In Fig. 4.12 and 4.13 we compare the performance of the link-based and the node-based schemes. The reduction in the redundancy is evident when we apply the node-based protection mechanism. In particular, a 7% reduction in wavelength redundancy is achieved by switching from a link-based to a node-based scheme when using our proposed link metrics. Observed the lowest two curves in Fig. 4.12. This validates our intuition that a relaxation on the destination nodes allows for the more diversified search for the best protection paths, which leads to a better utilization of the “network-wide” sharing potentials.

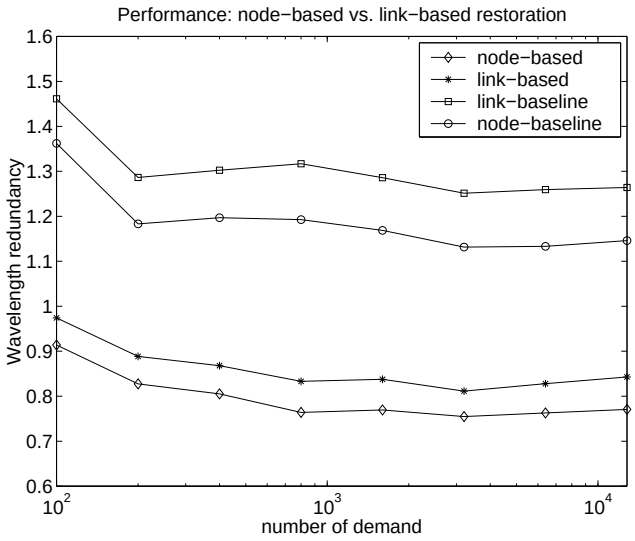

Figure 4.13: Link-based vs. node-based, NSFnet

4.5 Summary and discussion

In this chapter we explored a novel design for routing algorithms that aims to provide efficient failure protection in WDM networks. The key idea of the "bucket"-based link metrics can be applied to a range of restoration mechanisms, including link-based, node-based, or path-based approaches. We evaluated the effectiveness of this proposal via simulation and the results are promising. In the future we will look into the various issues that may impact the performance of our scheme. In particular, we intend to explore possible ways of improving the coordination of the protection path computation, probably among the protection requests that are initiated by the same working path. The trade-off between link-based and path-based protection schemes is theoretically interesting and practically useful. The benefit of joint working/protection path design, or the lack thereof, has not been studied here and warrants further investigation.

Our work in this chapter mainly focused on provisioning protection wavelengths in WDM networks. We assume the WDM network can provide fully dynamic reconfigura-

tion such as light-path switch-over upon a failure event. This may or may not be feasible in practice. Indeed, it is often the case that there exists a number of *overlay/virtual* networks running higher-layer protocols such as SONET/ATM/IP/MPLS on top of physical WDM infrastructure. These higher-layer networks may be equipped with their own failure recovery mechanisms. From a protection/restoration's perspective this is beneficial in that certain failures can be captured and dealt with in multiple layers and thus are unlikely to stay unattended for long and cause significant performance degradation. However, without further coordination the failures might "propagate" up the protocol stack, *e.g.*, a fiber cut at the WDM layer can cause multiple link failures at the IP/MPLS layer, since a WDM link might be mapped into multiple links at the overlay layers. To alleviate this problem, we need to either make lower level failures invisible to higher level networks[20], or jointly design network routing at both higher and lower level networks [14], or design network overlays so that the possibility of failure propagation is eliminated[10]. In the specific context of the work presented in this chapter, *i.e.*, provisioning resource for failure protection, this amounts to a need to account for multiple failures/alarms if the protection is considered at a certain overlay layer. The notion of bucket metrics is generic since each bucket corresponds to a failure event, not necessarily a specific link/node/path. Hence our approach can be adapted to the protection of any type of failure combinations. Evidently, it is not a trivial task to identify the set of failures that might happen and needs to be protected against in a layered network, and this warrants further research.

In this chapter we establish the protection paths as the working path is constructed, using the current resource sharing information expressed in the form of buckets. When the connections finish their sojourn the corresponding light paths need to be torn down and the corresponding working and protection resources will be returned to the network. In a dynamic environment where connections come and go the resource reserved for protection, *i.e.*, the height of the highest bucket, varies over time. A protection path that is "free-riding" at its setup time, *i.e.*, the one which need not reserve wavelength on a given link it

traverses since it can share the protection resource with other ongoing protection requests, might become the only one reserving the protection resource when other protection paths are torn down. Hence it seems beneficial to incorporate certain prediction capability into the protection provisioning mechanism so as to minimize the cost of protection for a given connection, not only at the time it arrives at the network, but *during its sojourn in the network*. The predictive mechanism suggested in the previous chapter, or other simple schemes accounting for the change in the sharing potentials at a given link, might help in this exploration.

Chapter 5

Conclusions

In this dissertation we study dynamic network routing problems. We take a flow level perspective of the network traffic and design routing schemes to achieve various network-wide and/or user-centric performance objectives. In particular, we consider routing as a dynamic decision-making process that not only responds to but also impacts the network condition. A well-devised routing scheme should try to satisfy an incoming user's QoS demand and limit the negative impact on the performance of the ongoing and future demands. To this end we identify a number of distinct operating regimes in which routing decisions, user demands and network states interact. We propose routing schemes based on the characteristics of these operating regimes.

In an operating regime where traffic demands are bursty and network states fluctuate in a highly dynamic fashion, we examine the performance of the dynamic single path routing scheme based on advertised link load information. We observe that although a dynamic routing scheme should respond to the advertised link states in order to avoid network congestion, an unwise reliance on outdated link states will lead to degraded network performance. We propose to address this problem by using a *dynamic multi-path routing scheme*, which disperses the traffic flows between a source and a destination over *a set of* least congested paths. We analyze the critical issue of how to select the set of paths over which to disperse traffic, and provide insights on the performance of this scheme in a mesh network.

We find our dynamic multi-path routing scheme is quite robust to various operating parameters and offers consistent performance improvement over a baseline single path routing scheme. Our finding, along with recent work in [66, 44] that aim at approximating optimal routing performance[11, 35], confirms that it is feasible to use dynamic multi-path routing scheme to improve routing performance when network states are dynamic and traffic demands are bursty.

In a network environment where network states and user demands evolve on a modest timescale, we design routing schemes that jointly consider flow properties, such as flow holding times, and network properties, such as the average link loads and mean reversion for the load dynamics. We show that in some cases we can improve both the network-wide and the user-perceived performance. The improvements are due to: (1) the predictability of network state, thus one can devise routing mechanisms that exploit it to benefit system performance, and (2) the advantages that can be gleaned by designing routing schemes that improve users' performance during their sojourn in the network, rather than at the time of arrival. We realize these ideas by modeling the link load dynamics and routing incoming flows so that during their sojourn in the network they see minimal average loads. As a result flows not only experience good performance, but also deliver least negative impact on the performance of other ongoing (and future) flows. By simulation we show that our routing scheme leads to a fair *and* efficient sharing of the network resources.

Our design strategy for protection path routing is quite different from the working path routing, both in terms of its objectives and algorithms. For protection routing the objective is to provide 100% failure protection so that when certain failures occur there will be enough spare capacity to allow for the re-routing of the traffic with minimal service disruption. This requires protection paths to be link or path disjoint from the associated working paths, therefore limiting the design space of the protection routing. We observe that a link-based protection path is restricted in selecting its "detours" around failure links, but it can share the protection resources with protection paths for other links. We identify

a routing metric that captures the sharing potential in the network and represent it by a set of “buckets” maintained at each link in the network. This representation, and its associated routing algorithms, are shown to result in protection paths with minimal marginal cost, and significantly reduce the number of wavelengths needed for failure provisioning. We note that this link metric is also applicable to path-based and node-based protection/restoration, and may be extended to account for not only the instantaneous marginal cost of protection, but also the “time-averaged” cost of sharing protection resources in the network.

Bibliography

[1] M. Alanyali and B. Hajek. On simple algorithms for dynamic load balancing. In *Proc. IEEE INFOCOM*, pages 230–238, 1995.

[2] J. Anderson, B. Doshi, S. Dravida, and P. Harshavaradhana. Fast restoration of ATM networks. *IEEE JSAC*, 12(1):128–138, 1994.

[3] G. Apostolopoulos, R. Guérin, and S. Tripathi. Quality of service based routing: A performance perspective. In *Proc. ACM Sigcomm*, pages 17–28, 1998.

[4] P. Bonenfant and A. Rodriguez-Moral. Optical data networking. *IEEE Comm. Magazine*, Vol. 38 No. 3:63–70, 2000.

[5] A. Brooke, D. Kendrick, A. Meeraus, and R. Raman. *GAMS language guide*. GAMS Development Corporation, 1997.

[6] R. Callon, N. Feldman, A. Fredette, G. Swallow, and A. Viswanathan. A framework for multiprotocol label switching. *IETF Internet Draft, draft-ietf-mpls-framework-03.txt*, June 1999.

[7] R. Coltun. The OSPF opaque LSA option. *RFC 2370*, 1998.

[8] C. Courcoubetis, A. Dimakis, and M. Reiman. Providing bandwidth guarantees over a best-effort network: call-admission and pricing. In *Proc. IEEE Infocom*, pages 459–467, 2001.

[9] E. Crawley, R. Nair, B. Jajagopalan, and H. Sandick. A framework for qos-based routing in the internet. *RFC 2386*, August 1998.

[10] O. Crochat, J. Le Boudec, and O. Gerstel. Protection interoperability for WDM optical networks. *IEEE/ACM Transactions on Networking*, 8(3):384–395, 2000.

[11] D.Bertsekas and R. Gallager. *Data Networks*. Prentice Hall, New Jersey, 1992.

[12] B. T. Doshi, S. Dravida, P. Harshavardhana, O. Hauser, and Y. Wang. Optical network design and restoration. *Bell Labs Technical Journal*, pages 58–84, Jan. - March 1999.

[13] D. Awduche *et. al.*. Multi-protocol lambda switching: Combining MPLS traffic engineering control with optical crossconnects (draft-awduche-mpls-te-optical-02.txt). *work in progress, Internet Draft*, July 2000.

[14] D. Awduche *et. al.*. Requirements for traffic engineering over MPLS. *IETF, RFC 2702*, Sept. 1999.

[15] G. Berstein *et. al.*. Optical domain service interconnect (ODSI) functional specification (version 1.4). *work in progress, ODSI Draft*, August, 2000.

[16] T. Chujo *et. al.*. The design and simulation of an intelligent transport network with distributed control. *Network Operations Management Symposium*, 1990.

[17] W.D. Grover *et. al.*. Development and performance verification of a distributed asynchronous protocol for real-time network restoration. *IEEE JSAC*, 9(1):112–125, 1991.

[18] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Networking*, 1(4):397–413, Aug. 1993.

[19] ATM Forum. P-NNI draft specification. Technical report, March 1995.

[20] B. Gavish and I. Neuman. Routing in a network with unreliable components. *IEEE Transactions on Communications*, 40:1248–1257, July 1992.

[21] N. Ghani, S. Dixit, and T.S. Wang. On IP-over-WDM integration. *IEEE Comm. Magazine*, Vol. 38 No. 3:72–84, 2000.

[22] R. Gibbens. Dynamic routing in fully connected networks. *IMA Journal of Mathematical Control and Information*, 7:77–111, 1990.

[23] R. Gibbens and F.P. Kelly. Resource pricing and the evolution of congestion control. *Automatica*, pages 1969–1985, 1999.

[24] R.J. Gibbens, F.P Kelly, and S.R.E. Turner. Dynamic routing in multiparented networks. *IEEE/ACM Trans. Networking*, 1(2):261–70, 1993.

[25] André Girard. *Routing and Dimensioning in Circuit-Switched Networks*. Addison-Wesley, Reading, MA, 1990.

[26] A. Greenberg, G. Hjalmtysson, and J. Yates. Smart routers-simple optics: A network architecture for IP over WDM. *Optical Fiber Conference*, 2000.

[27] W.D. Grover. The self-healing networks: a fast distributed restoration technique for networks using digital cross-connect machines. In *Proc. IEEE Globecom*, pages 1090–1095, 1987.

[28] E. Gustafsson and G. Karlsson. A literature survey on traffic dispersion. *IEEE Network*, pages 28–36, March/April 1997.

[29] L. Hafer. *bonsaiG User's Manual*. School of Computing Science, Simon Fraser University, 1997.

[30] R. R. Iraschko and W. D. Grover. A highly efficient path-restoration protocol for management of optical network transport integrity. *IEEE JSAC*, V. 18. No. 5:779–794, 2000.

[31] R.R. Iraschko, M.H. MacGregor, and W.D. Grover. Optimal capacity placement for path restoration in mesh surviviable networks. *IEEE ICC*, V. 18. No. 5, 1996.

[32] R.R. Irashoko, W.D. Grover, and M.H. MacGreger. A distributed real-time path restoration protocol with performance close to contralized multi-commodity maxflow. In *Proc. 1st Intl. Workshop on Design of Reliable Commun. Networks*, May 1998.

[33] V. Jacobson. Congestion avoidance and control. In *Proc. ACM Sigcomm*, pages 314–329, Aug. 1988.

[34] R. Kawamura, K. Sata, and I. Tokizawa. Self-healing ATM networks based on virtual path concept. *IEEE JSAC*, 12(1):120–127, 1994.

[35] F.P. Kelly. Loss networks. *Ann. Appl. Prob.*, 1:317–378, 1991.

[36] F.P. Kelly. Network routing. *Proc. R. Soc. Lond. A*, 337:343–367, 1991.

[37] F.P. Kelly. Notes on effective bandwidths. In F.P. Kelly, S. Zachary, and I.B. Ziedins, editors, *Stochastic Networks: Theory and Applications*, pages 141–168. Oxford University Press, 1996.

[38] Murali Kodialam and T.V. Lakshman. Dynamic routing of restorable bandwidth guaranteed tunnels using aggregated network resource usage information. In *Proc. IEEE Infocom*, pages 902–911, 2000.

[39] Murali Kodialam and T.V. Lakshman. Minimum interference routing with applications to MPLS traffic engineering. In *Proc. IEEE Infocom*, pages 884–893, 2000.

[40] S.A. Lippman. Applying a new device in the optimization of exponential queuing systems. *Operations Research*, 23(4):687–710, 1975.

[41] Q. Ma and P. Steenkiste. On path selection for traffic with bandwidth guarantees. In *Fifth IEEE International Conference on Network Protocols*, 1997.

[42] Q. Ma, P. Steenkiste, and H. Zhang. Routing high-bandwidth traffic in max-min fair share networks. In *Proc. ACM Sigcomm*, pages 206–217, 1996.

[43] A. Mandelbaum, W.A. Massey, and M.I. Reiman. Strong approximations for markovian service networks. *Queueing Systems*, 30:149–201, 1998.

[44] Vladimir Marbukh. A framework for combining long and short time-scale traffic engineering for MPLS. In *6th Informs Telecommunication Conference*, 2002.

[45] L. Massoulié and J. Roberts. Bandwidth sharing: objectives and algorithms. In *Proc. IEEE Infocom*, pages 1395–1403, 1999.

[46] N.F. Maxemchuk. Dispersity routing. In *Proc. ICC'75*, pages 41.10–41.13, June 1975.

[47] N.F. Maxemchuk. Dispersity routing on ATM networks. In *Proc. IEEE Infocom*, volume 1, pages 347–357, 1993.

[48] Steve McCanne. *Scalable Compression and Transmission of Internet Multicast Video*. Ph.D. Thesis, University of California at Berkeley, 1996.

[49] Debasis Mitra, John A. Morrison, and K. G. Ramakrishnan. ATM network design and optimization: A multirate loss network framework. *IEEE/ACM Transactions on Networking*, 4(4):531–543, August 1996.

[50] K. Murakami and H.S. Kim. Optimal capacity and flow assignment for self-healing ATM networks based on line and end-to-end restoration. *IEEE/ACM Transactions on Networking*, 6(2):207–221, 1998.

[51] S. Nelakuditi, Z. Zhang, and R.P. Tsang. Adaptive proportional routing: A localized QoS routing approach. In *Proc. IEEE Infocom*, pages 1566–1575, 2000.

[52] V. Nguyen. On the optimality of trunk reservation in overflow processes. *Probability in the Engineering and Informational Sciences*, 5:369–390, 1991.

[53] S. Plotkin. Competitive routing of virtual circuit in ATM networks. *IEEE JSAC*, 13(6):1128–1136, August 1995.

[54] Bellcore Special Report. Digital cross-connect systems in transport network survivability. *SR-NWT-002514*, issue 1, 1993.

[55] R.Gibbens, P. Hunt, and F. Kelly. Bistability in a communications network. In G. Grimmett, editor, *Disorder in Physical Systems*, pages 113–128. Claredon Press, 1990.

[56] H. Sakauchi, Y. Nishimura, and S. Hasegawa. A self-healing network with an economical spare-channel assignment. In *Proc. IEEE Globecom*, pages 438–443, 1990.

[57] A. Shaikh, J. Rexford, and K. Shin. Load-sensitive routing of long-lived IP flows. In *Proc. ACM Sigcomm*, volume 29, 1999.

[58] Scott Shenker. Fundamental design issues for the future internet. *IEEE J. Select. Areas Commun.*, 13(7):1176–88, Sept. 1995.

[59] S. Sibal and A. DeSimone. Controlling alternate routing in general-mesh packet flow networks. *Proc. of SIGCOMM94*, pages 168–179, 1994.

[60] I. Stoica and H. Zhang. LIRA: an approach for service differentiation in the internet. In *NOSSDAV*, pages 115–128, 1998.

[61] K. Struyve, P. Demeester, L. Nederlof, and L. Van Hauwermeiren. Design of distributed restoration algorithms for ATM meshed networks. In *Proc. IEEE 3rd Symp. on Commun. and vehicular tochnology*, pages 128–135, Oct. 1995.

[62] C.-F. Su and G. de Veciana. On statistical multiplexing, traffic mixes and VP management. In *Proc. IEEE Infocom*, 1998.

[63] S.R.E. Turner. The effect of increasing routing choice on resource pooling. *Probability in the Engineering and Informational Sciences*, 12:109–124, 1998.

[64] C. Villamizar. MPLS Optimized Multipath (MPLS-OMP). *Internet Draft*, Nov. 1998.

[65] C. Villamizar. OSPF Optimized Multipath (OSPF-OMP). *Internet Draft*, Feb. 1999.

[66] S. Vutukury and J.J. Garcia-Luna-Aceves. A simple approximation to minimum-delay routing. In *Proc. ACM Sigcomm*, pages 227–238, 1999.

[67] Z. Wang and J. Crowcroft. Analysis of shortest-path routing algorithms in a dynamic network environment. *ACM Computer Communication Review*, 22(2):63–71, April 1992.

[68] Z. Wang and J. Crowcroft. Quality-of-service routing for supporting multimedia applications. *IEEE JSAC*, 14(7):1228–35, Sept. 1996.

[69] T.H. Wu. A passive protected self-healing mesh network architecture and applications. *IEEE/ACM Trans. Networking*, 2(1):40–52, 1994.

[70] T.H. Wu. Emerging technologies for fiber network survivability. *IEEE Comm. Magazine*, pages 58–74, Feb.,1995.

[71] C.H. Yang and S. Hasegawa. Fitness: Failure immunization technology for network services survivability. In *Proc. IEEE Globecom*, pages 1549–1554, 1988.

[72] S. Yang and G. de Veciana. Size based adaptive bandwidth allocation: Optimizing theQoS for elastic flows. In *Proc. IEEE Infocom*, 2002.

[73] J.Y. Yen. Finding the k shortest loopless paths in a network. *Management Science*, 17:712–716, 1971.

Vita

Xun Su was born in Leshan, China on May 5, 1972, the son of Yunshang Zheng and Guangping Su. He graduated with a BSEE from University of Electronic Science and Technology of China in 1992. He earned his MSEE from Southeast University, China in 1995, with a focus on Computer Vision and Pattern Recognition. He entered University of Texas at Austin in September, 1996 and joined Dr. Gustavo de Veciana's networking research group in Spring 1998. He concluded his research at UT Austin in August 2002 with a focus on network routing algorithms and obtained a PhD in Electrical Engineering.

Permanent Address: 7401 North Lamar, #314, Austin TX. 78752

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.