In compliance with the
Canadian Privacy Legislation
some supporting forms
may have been removed from
this dissertation.

While these forms may be included
in the document page count,
their removal does not represent
any loss of content from the dissertation.

University of Alberta

# Scalable Quality of Service Routing

by

Yanxia Jia 

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Department of Computing Science

Edmonton, Alberta
Fall 2003

*Your file Votre référence*
*ISBN: 0-612-87999-2*
*Our file Notre référence*
*ISBN: 0-612-87999-2*

University of Alberta

Library Release Form

**Name of Author**: Yanxia Jia

**Title of Thesis**: Scalable Quality of Service Routing

**Degree**: Doctor of Philosophy

**Year this Degree Granted**: 2003



Date: July 22, 2003

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Scalable Quality of Service Routing** submitted by Yanxia Jia in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Dr. Michael H. MacGregor

Date: July 22, 2003

To my loving family

# Abstract

Quality of Service (QoS) Routing is concerned with the problem of finding paths to satisfy connections' QoS requirements, e.g., bandwidth, delay, jitter, packet loss, as well as to efficiently use network resources. One prominent problem with QoS routing is scalability. In this thesis, we approach the issue of scalability by tackling the following problems: QoS routing with inaccurate information, cost–effective routing information dissemination, and deflection routing.

Routing inaccuracy is tightly related to the scalability problem because scalable solutions are primarily aimed at reducing the amount of link state exchange, which directly causes imprecise routing information. Furthermore, some QoS metrics are not accurate, e.g. link delays. We deal with the inaccuracy problem using multi–path routing. We propose a family of routing construction and selection strategies to compensate for the inaccurate information. Our results show that among the proposed algorithms, the hop–based one provides the best performance. Compared with the single–path routing and other popular methods, our hop–based scheme shows superior capabilities in dealing with inaccurate information. Also examining the costs and benefits demonstrates that the proposed multi–path routing schemes are scalable solutions.

Confronted with the redundancy and blindness of the conventional flooding scheme, we propose a set of routing information dissemination schemes, i.e., the distance–based link state update schemes. The experiments show that the

proposed mechanisms offer comparable performance to flooding while greatly reducing the update overhead. Then based on the analysis of the reason behind large update periods, we propose to qualitatively separate the content of updates. We conclude that the link state update frequency is not the only factor that influences routing performance, and with large update periods, the nature of the link state information also has a dramatic impact on performance. Based on these observations, we can further improve the effectiveness of routing information distribution.

Finally, deviating from the traditional connection–based QoS provisioning model, we re–approach the multi–path routing problem without depending on fixed path sets. Under the deflection model, we investigate the impact of multi–path routing on real–time delay jitter sensitive voice traffic. Using multiple paths for packets inside a session brings about the issue of packet mis–ordering in the network, and therefore assembly buffers are needed at the destination; on the other hand, the lossless feature of deflection routing allows us to reduce the buffer space at the intermediate routers. Consequently, we address the trade–off of buffer resource allocation between intermediate routers and destination hosts. We also conduct a comparison between the proposed lossless deflection strategy and the conventional shortest path strategy. Our results show that, from the global viewpoint of network resource management, deflection–base routing is a preferable choice.

# Acknowledgements

I would like to express my great gratitude to my supervisors, Dr. Pawel Gburzynski and Dr. Ioanis Nikolaidis, for their invaluable guidance, support and encouragement in the past several years. They have been good role models for me in many ways, and it has been such a great experience for me to work with them. Without their consistent support, this thesis would not have been possible.

Also, I'm grateful to Dr. Janelle Harms and Dr. Michael MacGregor for their general help and support during my stay at the University of Alberta. I am very thankful for the committee members, Dr. Jacek Ilow, Dr. Witold A. Krzymien and Dr. Michael MacGregor, for reviewing my thesis.

Special thanks to my husband for sharing the happiness and sadness with me and for always being there for me; I am in great debt to my parents for their unfailing love and support.

The communication network group deserves my sincere thanks. Those stimulating and insightful discussions and fun moments will be unforgettable memories. My graduate study experience would not have been so fruitful and enjoyable without my wonderful group.

I would like to thank Dr. Catherine Descheneau for her kind support and Ms. Edith Drummond for the excellent graduate program work. I am also thankful for Mr. Steve Sutphen for his help.

Last but not least, I would like to thank the Computing Science Department for providing such a wonderful environment for me to grow professionally and personally.

# Contents

# List of Figures

# List of Symbols

| | |
|---|---|
| ABR | Available Bit Rate |
| AF | Assured Forwarding |
| AS | Autonomous Systems |
| ATM | Asynchronous Transfer Mode |
| BGP | Border Gateway Protocol |
| BKS | Best–$K$–Shortest |
| BKW | Best–$K$–Widest |
| BN | Bad News |
| CBR | Constant Bit Rate, Constraint-Based Routing |
| Cov | Coefficient of Variation |
| DA | Dynamic Alternative |
| DF | DeFlection Routing |
| DFR | Deterministic Frequency Reduction |
| DiffServ | Differentiated Services |
| DSCP | DiffServ Codepoint |
| ECMP | Equal–Cost–Multi–Path |
| IGP | Interior Gateway Protocol |
| EF | Expedited Forwarding |
| EGP | Exterior Gateway Protocol |
| EIGRP | Enhanced Interior Gateway Routing Protocol |
| FC | Frequency Coefficiency |
| GN | Good News |
| GBN | Good and Bad News |
| HDT | Hop/Distance Threshold |
| IETF | Internet Engineering Task Force |
| IntServ | Integrated Services |
| IS–IS | Intermediate System–Intermediate System |
| ISP | Internet Service Provider |
| LSA | Link State Advertisements |
| LSP | Label Switched Path |
| LSUP | Link State Update Period |
| OSPF | Open Shortest Path First |
| PLS | Periodic Link State only |
| PSR | Proportional Sticky Routing |
| MANET | Mobile Ad Hoc NETworks |

| | |
|---|---|
| MPLS | Multiple Protocol Label Switching |
| Nrt-VBR | Non–Real–Time Variable Bit Rate |
| PFR | Probabilistic Frequency Reduction |
| PHB | Per–Hop–Behavior |
| PNNI | Private Network–to–Network Interface |
| QoS | Quality of Service |
| RIP | Routing Information Protocol |
| RKW | Random–$K$–Widest |
| RSVP | ReSerVation Protocol |
| Rt-VBR | Real–Time Variable Bit Rate |
| SKW | Shortest–$K$–Widest |
| SP | Single Path routing |
| TE | Traffic Engineering |
| TOS | Type of Service |
| UBR | Unspecified Bit Rate |
| VCR | Virtual Circuit based Routing |
| VoIP | Voice over IP |
| WKS | Widest–$K$–Shortest |
| WS | Widest Shortest |

# Chapter 1

# Introduction

## 1.1 Preliminaries

This dissertation focuses on the scalability problem of *Quality of Service* (QoS) routing in the Internet. The diversification of Internet applications have raised new challenges to its service architecture. Observing that the existing best–effort model can not satisfy the diverse requirements of applications, researchers have proposed new QoS architectures, such as the *Integrated Services* (IntServ) and *Differentiated Services* (DiffServ) architectures. However, inharmonious with the proposed QoS architectures, current Internet routing mechanisms are still best–effort based, impeding the effectiveness of controlling QoS. Therefore, specialized QoS based routing appears to be an indispensable aspect in QoS provisioning.

In this chapter, we will first introduce routing basics in Section 1.2. Then in Section 1.3 and Section 1.4, we will review the currently existing QoS architectures and the routing deficiencies under the proposed architectures. In Section 1.5, we will touch upon some traffic engineering tools pertinent to QoS routing. Finally we will give the organization of the whole dissertation.

## 1.2 Routing Basics

### 1.2.1 Internet Unicast Routing Protocols Overview

The Internet is organized into areas named *Autonomous Systems* (AS). Each AS is composed of many routers working together under the control of a cer-

tain administrative entity, such as an *Internet Service Provider* (ISP), a university or a corporation. Based on the area in which routers exchange their information and cooperate to transport packets, routing protocols fall into two categories: the *Interior Gateway Protocol* (IGP) and the *Exterior Gateway Protocol* (EGP). IGP is an intra–AS routing protocol designed for routers within an AS to exchange routing information, whereas EGP is an inter–AS routing protocol designed for routers to exchange routing information between ASs [55].

Historically, IGP protocols fall into two categories: the *distance vector protocols* and the *link state protocols*. The *Routing Information Protocol* (RIP) [33, 51] and the *(Enhanced) Interior Gateway Routing Protocol* ((E)IGRP) are representatives of distance vector routing protocols, while the *Open Shortest Path First*(OSPF) protocol and the *Intermediate System–Intermediate System* (IS–IS) protocol are typical link state routing protocols. A typical example of EGP is the *Border Gateway Protocol* (BGP), a path vector protocol. The content of BGP is beyond the scope of this dissertation, therefore in the following, we will only present RIP and OSPF, the representatives of IGP protocols.

### 1.2.2 Distance Vector and Link State Routing Protocols

A distance vector protocol employs a *distributed processing model* [55], under which route computation is distributed on the whole network. In the distance vector routing mechanisms, "distance" refers to the path construction metrics, i.e., the path distance in number of hops, and "vector" means the routing table. Specifically, all the routers cooperate to find the best path to each destination. In the following, we will present the routing procedure of RIP, a canonical example of distance vector protocols.

- Each RIP router maintains a routing table, each entry of which indicates the path distance to a certain destination and the corresponding next hop.

- Every 30 seconds, every RIP router broadcasts its routing table to all of its neighbors.

- Upon receiving the routing table update from its neighbor, $N$, a RIP router, $R$, checks all the destinations of the update and finds out the new path distance from itself to the destinations via $N$ (by adding the distance from itself to $N$ to the distance from $N$ to a certain destination). If the new path distance is shorter than the one in its current routing table, then the next hop of $R$ is updated to its neighbor $N$.

The advantage of RIP is its simplicity, but it also has drawbacks. First, it is only suitable for moderate sized networks, i.e., the largest hop length is 15. Second, in case of network failures, it takes a long time for RIP to converge. This is referred to as *counting to infinity* and is the reason why the maximum path length is limited to a very small number. Schemes such as *split horizon* and *triggered updates* [33] have been proposed to deal with counting to infinity, but they only work in limited situations. Third, RIP uses a single fixed metric to calculate paths, and therefore can not deal with situations where paths need to be chosen based on real–time parameters, such as reliability or load.

A link state protocol such as OSPF employs a *distributed database model* [55]. The basic idea is as follows:

- Each router maintains a *link state database*, a topology map of the whole network. Based on the database, each router conducts route computation using Dijkstra's shortest path algorithm.

- Every 30 minutes, each router sends *Link State Advertisements*(LSA), the topology information of itself, to every other nodes through *reliable flooding*. In doing so, each router has an identical topology map of the network so that routing loops and "black hole" are avoided.

As a representative of link state protocols, OSPF provides more advanced features than the typical distance vector protocol RIP. For example:

- *fast convergence.* By flooding link state information, all the OSPF routers can immediately receive the updated status of the network.

- *more descriptive routing metrics.* Unlike RIP, which only uses the hop count as a metric, OSPF allows for a variety of link metrics, e.g., link

delay and costs, etc. In addition, the value of the configurable link metric ranges from 1 to 65,535, eliminating the network diameter limitation of 15 hops in RIP and being able to support large networks.

- *supporting routing hierarchy.* OSPF provides a scalable routing solution by supporting routing hierarchy, while RIP only supports routing in "flat" networks.

- *supporting equal-cost-multi-paths.* OSPF supports multiple paths with equal costs, while RIP only supports single path routing. The former obviously increases the possibility of load balancing.

## 1.3 Quality of Service Architectures

The current standard IP-based Internet supports only *best effort* services. Under this service model, all the applications compete together for the limited network resources, such as buffers and bandwidth. As a result, applications can not predict their service quality, which depends on the congestion situation on the network. For example, if the buffer at a certain port is full, then the packets going through that port will have to be dropped. If there is not enough bandwidth available on a certain link, then packets will have to wait in the queue and experience long queuing delays.

With the overwhelming growth of the Internet, various new types of applications have gained increasing importance, e.g., IP telephony, video conferences, and on-line games, etc. The emerging applications have posed new challenges to the Internet community. In particular, the diversification of application types calls for service differentiation. For example, the quality of *Voice over IP* (VoIP) and video conferencing applications are extremely dependent on delay and delay jitter (the delay variation), while for file transfer applications, the most important quality metric is the loss rate. Moreover, satisfactory services require resource assurance. For instance, IP telephony has stringent delay requirements: if the delay between two conversation party exceeds 0.5 seconds, then the perceived conversation quality would be unac-

ceptable. Confronted with the diverse requirements of emerging applications, the current IP–based network, unfortunately, does not meet the challenges, which motivates the research on *quality of service* (QoS). Quality of Service refers to "the capability to provide resource assurance and service differentiation in a network" [69]. In the following sections, we will review the major QoS architectures.

### 1.3.1 The Asynchronous Transfer Mode (ATM)

The Asynchronous Transfer Mode is a technology based on fast packet switching of small fixed sized packets called cells. It is designed to provide a high–speed low–delay multiplexing and switching network to support diverse user traffic, such as voice, video, and data applications [68]. Its underlying feature is QoS support, which is facilitated by its connection–oriented nature and negotiation phase, i.e., contract driving connection setup. ATM supports service differentiation by defining five service categories: the *Constant Bit Rate* (CBR), the *Real–Time Variable Bit Rate* (rt–VBR), the *non–Real–Time Variable Bit Rate* (nrt–VBR), the *Available Bit Rate* (ABR), and the *Unspecified Bit Rate* (UBR). The CBR service is used for traffic with very strict bandwidth requirements, e.g., video on demand applications. The network offers constant bandwidth and minimum delay, delay variation and loss rate. The rt–VBR service is suitable for real–time applications, such as real–time video conferencing, requiring tight delay and delay variation, but not exhibiting the fixed bit rate of CBR. The nrt–VBR service is used for bursty data that is not sensitive to delay variation, e.g., multimedia E–mails. The UBR service is a best effort service with no real guarantees, and an example application is file transfers. The ABR service is intended for applications that can adapt their rates based on the feedback from the network. When an ABR connection is established, the user specifies a maximum required bandwidth and a minimum usable bandwidth. The bandwidth available from the network may vary, but is never less than the minimum bandwidth [27].

Despite its high–bandwidth promise and QoS features, ATM has not been widely used as expected because while ATM was in the process of standard-

ization, the emerging Gigabit Ethernet technology and QoS capabilities of IP networks reduced the attractiveness of ATM. In addition, ATM is too complicated and expensive to employ, making customers reluctant to adopt it once cost–effective solutions became available.

### 1.3.2 The Integrated Services/RSVP Model

The Integrated Services (IntServ) [15] framework supports the transport of audio, video, real–time, and classical data traffic within a single network infrastructure. The architecture provides the ability for applications to choose among multiple controlled levels of delivery services for their data packets. Currently, two types of services are defined — the *Controlled–Load Service* [71] and the *Guaranteed Service* [67]. The former is for adaptive tolerant real–time traffic (i.e., traffic with loose delay requirements). It guarantees a QoS similar to that achievable by a best effort traffic in an unloaded network. The latter is for non–tolerate real–time applications with tight delay requirements. It guarantees assured level of bandwidth, mathematically bounded end–to–end delay and no loss for conforming traffic, which stays within the "expected capacity".

The IntServ model features per–flow resource reservation. This idea is similar to the virtual circuit in ATM, but the objective of IntServ is to preserve the datagram model of IP networks while supporting resource reservation for real–time applications, and the challenge is to integrate resource reservation into the existing Internet architecture [69].

In [15], Braden, Clark and Shenker proposed a reference framework (see Figure 1.1) to implement the IntServ model. Our work is based on the assumption of this framework. According to this framework, the IntServ model encompasses five components: the reservation setup protocol, the QoS routing agent, the admission control agent, the packet classifier, and the packet scheduler. The first three components are background functions, and the last two are functions in the forwarding path. According to the Integrated Service model, an application must depend on resource reservation to receive guaranteed service. Therefore, the reservation protocol is the key component of the background functions. But QoS routing and admission control is needed to

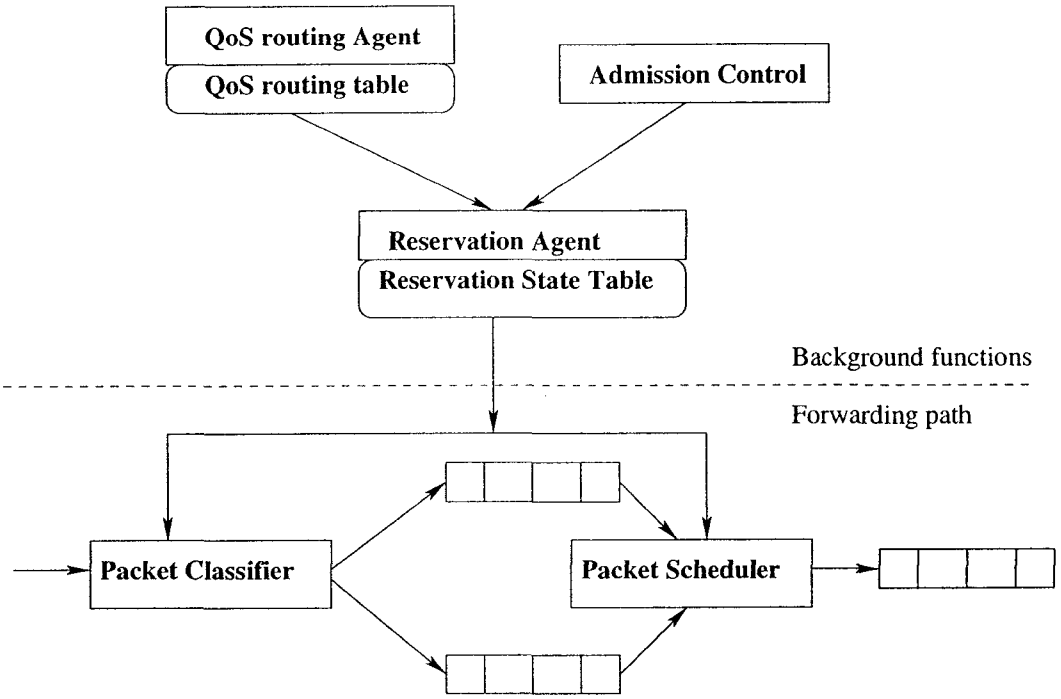


Figure 1.1: A Reference framework for Integrated Service model.

assist with the reservation setup. In particular, QoS routing is responsible for establishing the QoS paths, along which the reservation is to be made. During the procedure of reservation setup, the admission control agent at each node determines whether the resource request should be granted or rejected based on its knowledge about the resource availability. In the forwarding path, the classifier identifies each packet as belonging to a certain reserved flow and maps it to a certain service class. All the packets inside the same class are treated equally. Once a packet has been identified, the router will look up the reservation table to find the corresponding reservation state, and then the packet will be dispatched to the packet scheduler with the reservation state associated with the flow [69].

In the IntServ architecture, the *ReSerVation Protocol*(RSVP) [16] has been developed as the reservation setup protocol. According to [16], " The RSVP protocol is used by a host to request specific quality of service from the network for particular application data streams or flows. RSVP is also used by routers to deliver quality–of–service (QoS) requests to all nodes along the path(s) of the flows and to establish and maintain state to provide the requested service.

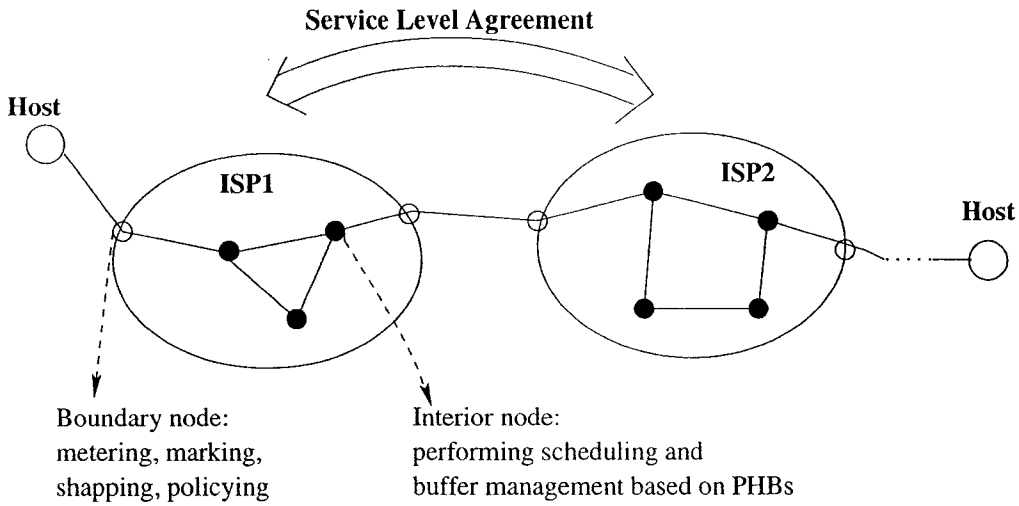


Figure 1.2: An example Differentiated Service network.

RSVP requests will generally result in resources being reserved in each node along the data path." To sum up, RSVP provides per–flow reservation support for the IntServ model, which implies accurate and fine–grained QoS provisioning. However, the per–flow state maintenance and per–flow processing at the routers also make the IntServ model suffer from scalability problems, which motivates further research into QoS models.

### 1.3.3 The Differentiated Services Model

The goal for the differentiated services (DiffServ) [11] model is to define a scalable service discrimination without the need for per–flow state and signaling at every hop. It pushes most of the state information to the edges of a network, while the core network only needs to perform fast forwarding. Traffic entering a network is classified and probably conditioned at the boundary of the network and assigned a *DiffServ codepoint* (DSCP), which is the first six bits in the *Type of Service* (TOS) byte in the IP header. Within the core of the network, packets are forwarded according to the *per–hop–behavior* (PHB) associated with the DSCP [11]. The DiffServ architecture "is composed of a number of functional elements implemented in network nodes, including a small set of per–hop forwarding behaviors, packet classification functions, and traffic conditioning functions including metering, marking, shaping and polic-

ing." Figure 1.2 depicts an example DiffServ network.

Two PHBs are defined in the DiffServ architecture: the *Assured Forwarding* (AF) and the *Expedited Forwarding* (EF). AF provides better than best–effort services. It specifies four forwarding classes, within each there are three drop priorities. EF provides minimum delay and jitter and is suitable for mission critical applications.

### 1.3.4 Refinement of the QoS Architecture: IntServ over DiffServ

As a summary to the above description, the IntServ/RSVP model enables per–flow quantifiable resource reservation along end–to–end data paths and the DiffServ model enables scalability across large networks. The two models can be characterized as representing a state–based and an (almost) stateless approach respectively. Owing to the per–flow state maintenance, the IntServ/RSVP model provides accurate service level but raises the scalability problem. In contrast, the DiffServ model possesses scaling properties because of being stateless, however, it introduces some degree of service level approximation. These two architecture are complementary in pursuit of end–to–end QoS. Based on this observation, further refinement of the QoS architecture is proposed [10], which applies the IntServ model end–to–end across a network containing one or more DiffServ regions. The IntServ is used as the architecture which allows applications to interact with the network, and DiffServ is used as the architecture to manage admission the network's resources [10]. As presented in [34]: "The architectural direction that appears to offer the most promising outcome for QoS is not one of universal adoption of a single architecture, but instead use a tailored approach where aggregated service elements are used in the core of a network where scalability is a major design objective and use per–flow service elements at the edge of the network where accuracy of the service response is a sustainable outcome."

## 1.4 Routing Deficiencies in QoS Architectures

Although capable in different ways to provide Quality of Service, both the IntServ/RSVP and the DiffServ architecture implicitly assume that "the shortest path" routing is used to route both best effort traffic and distinguished service traffic [34]. Up till now the most prevailing IGP protocols are shortest–path and destination–based routing, which possesses the inherent drawback of causing congestion and inefficient resource allocation. Specifically,

- In destination–based routing, the destination address is the only information used in forwarding a packet. Although this property enables highly scalable routing, it also limits the capability to influence the actual paths taken by packets. For example, it limits the capability to evenly distribute traffic among multiple paths.

- The shortest path might not be able to accommodate the flow's QoS request and request combination, e.g., bandwidth and loss rate, etc. However, alternative longer paths exist which can satisfy the requests.

- Traditional routing schemes provide limited support for alternate routing, which incurs inefficient resource usage on the network. That is, if the best path cannot admit a flow, the flow will be blocked even if some other, suboptimal, alternate paths exist. The Open Shortest Path First (OSPF) [55] includes an *Equal–Cost–Multi–Path* (ECMP) mechanism, which supports load splitting between equal–cost–multi–paths. However, since ECMP does not always exist, it cannot provide load splitting between multiple paths with different costs. Moreover, the exactly equal load share cannot be changed even if the load distribution over these paths has already been biased.

Therefore, the conventional shortest–path and destination–based routing does not provide QoS support, nor does it enable network–wide efficient resource utilization, which are essentially *Traffic Engineering* (TE) problems. The *Internet Engineering Task Force* (IETF) have introduced some new routing–

related traffic engineering technologies. The most important ones are *Constraint–based Routing* (CBR) and *Multiple Protocol Label Switching* (MPLS).

## 1.5 Traffic Engineering Tools: MPLS and CBR

Traffic engineering is the ability to control specific routes across a network to reduce congestion and improve the cost efficiency of carrying IP traffic. MPLS is strategically significant for traffic engineering. It simplifies and expedites the routing process by using labels to determine the next hop. Furthermore, MPLS also incorporates features from both the IntServ and DiffServ. For example, it allows bandwidth reservation to be made over a *Label Switched Path* (LSP) and support packet marking based on their priorities. More importantly, its explicit routes provide a mechanism for overriding the paths established by IP routing. For example, MPLS allows administrators to explicitly configure LSPs to send selected traffic (such as VoIP) along QoS routes. The LSPs which can satisfy the special requirements of traffic are computed using Constraint–based Routing, an important component of Traffic Engineering. Specifically, CBR computes a feasible network path based on a traffic description and a set of constraints. The constraints include bandwidth, hop count, delay, cost, and administrative attributes, such as bandwidth requirements, maximum hop count, and administrative policy requirements, etc. Its objective is to compute routes that meet QoS requirements, while efficiently utilizing the network resources.

For routers that use topology driven hop–by–hop IGPs, CBR can be incorporated in at least one of two ways [8]:

- *Overlay model*: by adding a CBR process which can co–exist with current IGPs [9].
- *IGP Extension model*: by extending the current IGPs, such as OSPF and IS–IS, to support constraint–based routing [3]. In this model, some enhancement is required to enable the IGP protocols perform CBR. For instance:

- *Content of LSA.* Apart from existing topology change LSAs, such as link down and restoration, more state information is needed, such as remaining bandwidth on the network links, which is indispensable for constraint–based routing calculation.
- *Frequency of LSA updates.* The Internet is a dynamic network environment, in which not only topology changes occur, but also the resource status (such as available link bandwidth) keeps changing at a rate much higher than that of the topology changes. Hence, LSA updates need to be flooded more frequently compared to normal IGPs. Compared with conventional routing computation, CBR computation is more dependent on this dynamic status, therefore accurate CBR calculation requires more frequent updates of the dynamic link state information.

## 1.6 Structure and Contribution of the Dissertation

In the previous sections, we have reviewed the working mechanisms of conventional IGP routing techniques, the current QoS architectures and the deficiency of routing under these architectures. In the following, we will describe the concept and challenges of QoS routing in Chapter 2 . The following three chapters focus on the scalability problem: in Chapter 3, we will review existing approaches to tackle the QoS routing problem with inaccurate information; in Chapter 4, a family of multiple path construction and selection schemes will be presented to deal with the routing inaccuracy problem; in Chapter 5, we will propose and evaluate a set of cost–effective routing information dissemination schemes as a further effort towards the scalability issue of QoS routing. Chapter 6 will focus on a connectionless multi–path routing paradigm, i.e., the deflection–based routing. We will primarily investigate the buffer trade–off problem associated with this routing model.

The contribution of the dissertation are as follows:

- First, we deal with the scalability problem of QoS routing from the

point view of routing algorithms. In particular, to compensate for routing inaccuracy, we propose a family of multiple path construction and selection algorithms, one category of which provides good performance with imprecise routing information. The simulation results show that the proposed algorithm outperforms the traditional single–path scheme, providing improved blocking rates and load balancing with lower routing costs. Moreover, compared with some popular QoS routing methods discussed in the literature, it offers better performance in an inaccurate environment [38, 39, 40].

- Second, we approach the scalability problem from the point view of routing information dissemination [40, 41]. Specifically, to reduce the redundancy and blindness of the current flooding mechanism without significantly degrading routing performance, we design selective routing information dissemination schemes which assign different update priorities to nodes based on their location information. Moreover, by differentiating between "good"(increase of available resource) and "bad" (decrease of available resource) updates, we observe a noticeable performance gap, which implies that the quality of link state dissemination can be improved by qualitatively separating link state updates.

- Finally, as an extension of our previous work of multi–path routing under a connection–based paradigm, we further explore the multi–path routing strategy under a connectionless paradigm, i.e., deflection routing. We examine the impact of deflection routing on real–time delay jitter sensitive voice traffic and address the trade–off of buffer resource allocation between intermediate routers and destination hosts [42]. We also conduct a comparison between the proposed lossless deflection strategy and the conventional shortest path strategy. Our results show that, from the global viewpoint of network resource management, deflection–base routing is a preferable choice. These results inspire our further thinking about the routing and congestion control problems of the Internet.

# Chapter 2

# QoS Routing

In the previous chapter, we have learned that QoS–based routing is a missing piece in the proposed QoS architectures and that some important QoS technologies, such as MPLS, need the support of constraint–based routing, which is a generalization of QoS routing. In this chapter, we will present the concept of QoS routing, how it is related to the previously covered QoS concepts, and the main issues in QoS routing.

## 2.1 What is Quality of Service Routing?

QoS routing is a routing mechanism that determines paths for flows based on knowledge about resource availability in the network and QoS requirements of the flows [22]. As implied in the definition, the objectives of QoS routing are as follows:

- *To meet the flow's QoS requirements.* Traditional routing algorithms are concerned with path reachability, length, and cost, etc, but cannot deal with diverse QoS requirements of a path, such as bandwidth, delay, jitter, and loss rate, etc.

- *To efficiently use network resources.* By taking the dynamic resource status into consideration, QoS routing can be designed to make efficient use of network resources, e.g., to avoid over–utilization and under–utilization of network resources and to balance the load of network traffic. Traditional shortest path based routing protocols do not take this factor into

consideration when making routing decisions.

## 2.2 QoS Routing and Other QoS Issues

### 2.2.1 QoS Routing vs. Resource Reservation

QoS routing and Resource Reservation are often used in conjunction. QoS routing serves as the routing mechanism needed by the Resource Reservation to find a path with QoS capabilities, but it cannot provide resource reservation by itself. Conversely, resource reservation includes resource request and reservation mechanism, but without determining the path to accommodate the request with QoS requirements [22].

### 2.2.2 QoS Routing vs. Traffic Engineering and Constraint–based Routing

As addressed above, one of the objectives of QoS routing is to efficiently use network resources, e.g., find multiple paths to balance traffic load. Therefore, QoS routing can be thought of as part of dynamic Traffic Engineering [29]. "It can also be seen that constraint based routing (including QoS routing), MPLS and DiffServ, all provide mechanisms (control capabilities) in support of Internet Traffic Engineering [7]."

CBR is the extension of QoS routing, and they share some common characteristics, such as satisfying the QoS requirements of the traffic and utilizing resources efficiently. Their difference exists in that CBR is defined in a broader sense as applicable to traffic trunks (flow aggregations) as well as individual flows, while QoS routing is individual flow oriented. Traffic trunks are considered to make "large" reservation with long holding time, which is the case for network core traffic aggregates. Besides, CBR extends QoS routing by dealing with non–QoS constraints like administrative policies.

## 2.3 Main Issues in QoS Routing

### 2.3.1 Introduction

As QoS requirements are added into routing schemes, many relevant problems come up. For instance, some major ones [22] are as follows:

- *QoS Metrics and computational complexity.* The combinations of different QoS metrics greatly impact the routing decision process. As we will see in Section 2.3.2, some of the combinations can be computationally hard problems. Therefore efficient heuristics in multi-constraint-based QoS routing computation are needed to reduce the complexity and reach good performance as well.

- *Routing overhead and scalability.* With QoS being introduced into routing protocols, network states become more dynamic from the viewpoint of the routing protocol. The reason is that, compared with conventional routing metrics, such as path hop counts and cost, etc, QoS routing parameters vary frequently, e.g., remaining bandwidth. As a result, more frequent routing information updates are needed, which unavoidably brings about sharp increase of routing overhead, especially when flooding [55] is used to exchange network updates. This increased overhead introduces scalability problems.

- *Routing performance objectives.* QoS routing does not merely mean routing a flow on any path that accommodates its QoS requirements. This simple approach ignores the resource allocation for the whole network and hence is likely to be ineffective. For example, as we will see in Chapter 3, given a set of eligible paths that can satisfy the QoS requirement of a connection, how to select a path for the connection significantly influence the resource allocation situation of the network and therefore the routing performance. Similar consideration of global resource allocation also exists in the telephone networks, e.g., *trunk reservation* [44]. The idea is to use alternative paths of a certain source-destination only

when they do not negatively influence the shortest direct paths for another source–destination pair. Other performance objectives also need to be considered, e.g., global throughput, load balancing, and oscillation avoidance, etc.

- *Routing decision granularity.* In descending order of granularity, routing can be destination–based, (as in the current Internet routing protocols), source–destination based, (as in QOSPF [77]), and flow–based, (as in the InteServ/RSVP architecture). Finer granularity schemes provide accurate routing decisions but heavy state overhead, while coarser granularity schemes trade routing accuracy for small state overhead. In this dissertation, we will use flow–based granularity, as most of the current QoS routing research [6, 47, 65] does.

In the following, we will mainly address the first two problems as they are the foundation of the feasibility of QoS routing.

### 2.3.2 QoS Metrics and Computation Complexity

QoS route computation can be very complicated because it involves multiple time–varying route metrics. According to [70], link metrics fall into three categories: the additive, the multiplicative and the concave. An additive metric (see (2.1) ) of a path is the summation of the metric of each single link on the path. Typical examples for the additive metrics are delays and monetary costs. A multiplicative metric (See (2.2)) of a path is one minus the product of one minus the metric of each single link on the path, e.g., the loss rate metric. A concave metric (See (2.3)) of a path , e.g., the available bandwidth, is the minimum metric of all the links along the path. We call that link with the minimum metric the bottleneck link. The following formulas depict the characteristic of the three kinds of metrics.

$$M_p = \sum_{l \in p} M_l \tag{2.1}$$

$$M_p = 1 - \prod_{l \in p} (1 - M_l) \tag{2.2}$$

$$M_p = min_{l \in p} M_l, \tag{2.3}$$

where $M_p$ is the metric of a path $p$ and $M_l$ is the metric of a link $l$ on the path. In QoS routing, it is not uncommon for the user to have a request involving multiple metrics. Unfortunately, the problem to compute a path with two or more of additive and multiplicative metrics is NP–hard [36, 70]. The formal description of the problem is as follows:

*Given $n$ additive metrics $A_i$ and integers $a_i$ separately, where $1 \leq i \leq n$, and $K$ multiplicative metrics $M_j$ and integers $m_j$ separately, where $1 \leq j \leq k$, the problem of deciding if there exists a path $p$ which satisfies*

1. $A_i^p \leq a_i$, $(2 \leq i \leq n)$, *or*
2. $M_j^p \leq m_j$, $(2 \leq j \leq k)$, *or*
3. $A_i^p \leq a_i$, $(1 \leq i \leq n)$ *and* $M_j^p \leq m_j$, $(1 \leq j \leq k)$

*is NP–hard.*

Specifically, if the metrics are two or more of delay, delay jitter, cost, and loss rate, etc, the route computation problem will be NP–hard [70]. Heuristic algorithms have been proposed to deal with this problem [36, 43, 45, 57, 64, 72], but this is not our focus. In this dissertation, we concentrate on the routing overhead and scalability problem of QoS routing.

### 2.3.3 Routing Overhead and Scalability

QoS routing encompasses two tasks: QoS–based route computation and network status propagation and collection. Correspondingly, the main overheads of QoS routing are computational overhead, storage overhead and protocol communication overhead [6]. Among all these cost factors, the protocol communication overhead is the dominant contributor [4, 6]. It is caused by link state information generation and propagation since most of the current QoS routing protocols are link state based [2, 6, 47, 66], whose characteristic is to frequently exchange link state information by flooding. The exponential growth of the number of Internet users implies a sharply increasing routing

protocol communication costs. Moreover, with flooding being the main protocol communication mechanism in current QoS routing schemes, each node sends out its state information to all the other nodes, even if the latter might never or seldom use the information. Obviously, a great deal of redundant information is generated in this procedure, which occupies a lot of network resources. The situation will deteriorate as network size increases. Furthermore, QoS routing requires more link state updates (LSU) than conventional routing does because, unlike traditional routing, which cares about infrequently changing connectivity information such as link up/down changes, QoS routing cares about frequently changing QoS metrics such as remaining bandwidth, and therefore requires more frequent updates to support accurate route calculation. Consequently, the protocol communication overhead makes scalability a key issue in QoS routing.

In [31], a taxonomy of routing scalability techniques is presented. Current techniques fall into two basic categories: techniques for reducing path computation and techniques for reducing routing updates. (See figure 2.1)

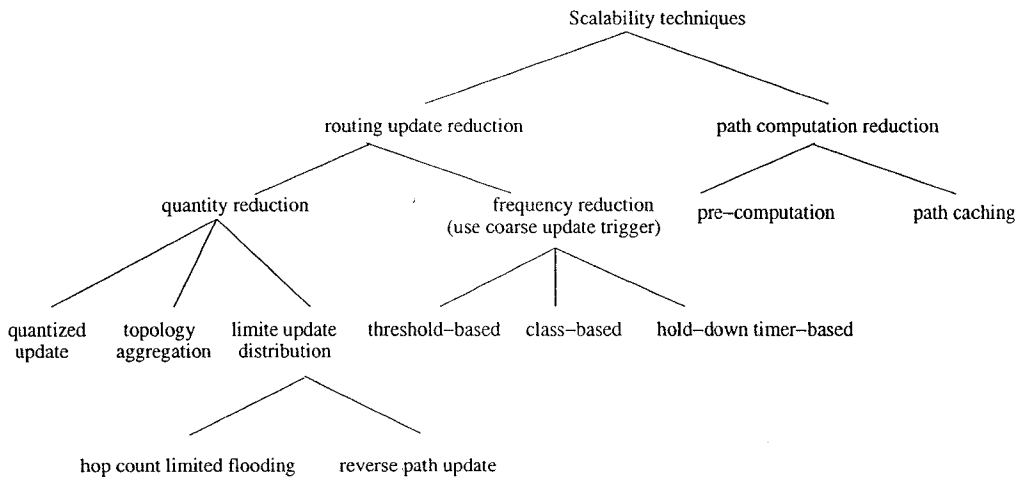


Figure 2.1: A taxonomy of scalability techniques

*Pre–computation* [14, 48, 61, 65] and *path caching* [60] are two well known approaches to reducing path computation. Path–computation means computing paths periodically or upon receipt of a network state update. It was shown [6] to be more cost–effective than *on–demand* computation (compute

for each request) although the former results in a small routing performance loss. Efforts to reduce routing updates include *quantity reduction* and *frequency reduction*. Quantity reduction aims to compress the update contents and to limit the update propagation range. Frequency reduction, as the name implies, reduces the frequency of link state updates. In the following, we will address the two techniques.

In the frequency reduction approach, update triggers [6] are used to determine when to propagate a link update. It is an effective technique [6] used by many QoS routing schemes [2, 6, 47, 66] to reduce link state update costs. Typically, there exist 3 categories [6], the *threshold–based*, the *class–based* and the *hold–down timer based*. The threshold–based trigger does not emit QoS LSU until link metric changes exceed a certain absolute or relative threshold. In the class–based scheme, link state is divided into several classes. Whenever a link state change runs across a class edge, an update is triggered. The hold–down timer based trigger conducts updates when the hold–down timer expires. Generally the hold–down timer is used in conjunction with the other two schemes to control the volume of updates when network state oscillates in a narrow range. Obviously the trade–off of the setup of this parameter lies in the update costs and routing accuracy.

The quantized update content approach means disseminating some quantized approximation to link state information instead of exact one. Obviously, the update information is not precise, but it reduces the amount of updates and also increases the number of equal cost paths, which avoids being stuck with a "bad" choice. [6]

The topology aggregation in ATM Private Network–to–Network Interface (PNNI) [26, 32, 46] is a typical example of compressing update content. According to the scheme, large networks are structured hierarchically by recursively grouping nodes into routing domains (See Figure 2.2). The topology information of each domain is represented in a "compact" manner. To the outside of a domain, the original topology information inside the domain is invisible. Instead, what can be seen is simplified topology information of the domain. Apparently, topology aggregation simplifies the representation of the

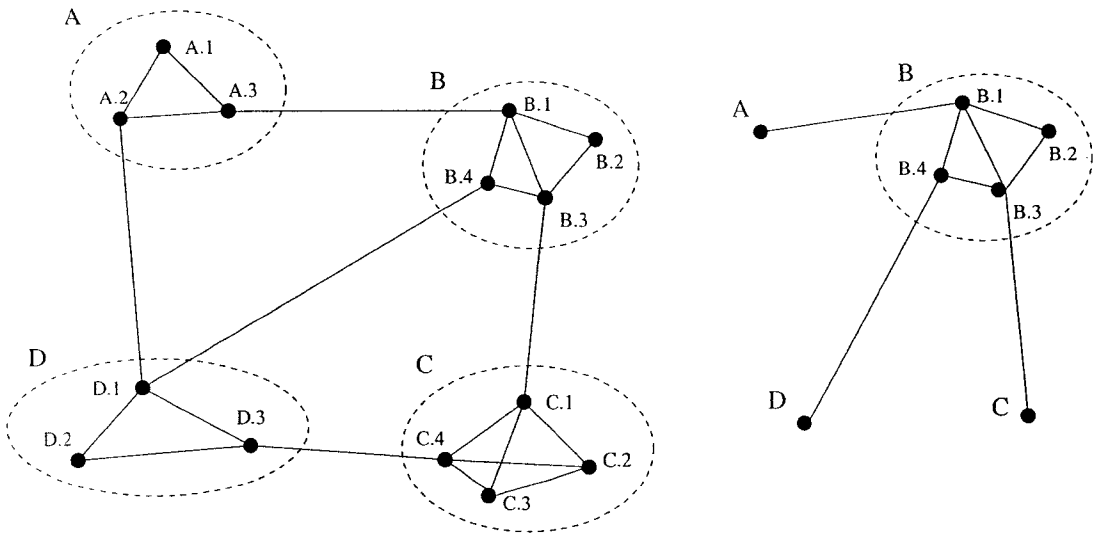


Figure 2.2: Topology Aggregation

network structure, reduces routing information exchange and cuts down on routing table sizes, and therefore it scales well.

Although PNNI [26] defines the idea of topology aggregation, it does not specify how to perform aggregation. Certain aggregation schemes have been proposed. Specifically, as presented in [46], the symmetric-node approach, the full mesh approach and the star approach are conventional methods for topology aggregation. Routing performance can be very different depending on the aggregation method. Hao [32] performed a systematic evaluation of the performance of different routing algorithms under topology aggregation and concluded that topology aggregation does not always have a negative impact on routing performance. Despite these observation, the results in [32] showed that performance degradation is not significant only in limited situations. The reason behind the performance deterioration is the inherent "lossy" characteristic of topology aggregation and the consequential non-optimal path selection. Besides, the problem becomes more complicated when multiple QoS constraints are involved [62]. For instance, if there exist two paths between two border nodes of a domain, with one path favoring bandwidth constraint

and the other favoring delay constraint, it is difficult to pick the best QoS pair. Open issues like this make topology aggregation a challenging problem.

Limited update distribution aims to reduce the flooding range or reduce global flooding to partial flooding. Two existing methods are hop count limited flooding and reverse path update [17]. Hop count limited flooding means distributing link state information in a scope whose radius is less than a certain number of hop counts. Reverse path update [17, 24] exploits the sparse characteristics of internetwork traffic [17], for example, routes originating from a domain are often destined for a certain set of domains or frequently pass through some transit domains. It assumes that the current active routes are still going to be active in the future. The idea is to propagate routing information updates only to the sources of the current active routes, which are recorded at each node and represented by installed setup state or by a cache of the most recent used source routes. The propagation is along the reverse direction of the active routes.

In summary, the aforementioned techniques deal with the scalability problem by reducing either the amount of link state exchange or path computation. These techniques have proved to be effective in reducing routing cost, however, the price paid is performance deterioration. Consequently, in the next chapter, we will address the inaccuracy tolerant QoS routing problem.

# Chapter 3

# Inaccuracy Tolerant QoS Routing

As presented in Chapter 2, the reduction of QoS routing overhead causes performance degradation. For example, with path pre–computation, some flows have to use obsolete routing table information. The performance of pre–computation is worse than that of on–demand computation, especially when the pre–computation period is large. Moreover, routing update reduction brings about stale state information (e.g., by infrequent updates) or approximated link states (e.g., by topology aggregation) on network nodes, which can cause a node to select a suboptimal path or a path that cannot accommodate the new connection. Aside from that, some QoS metrics per se are imprecise, e.g., the link delay, which is dependent on the dynamically changing traffic characteristics. Therefore, how to conduct QoS routing with inaccurate information is a crucial task for the QoS routing scalability problems. In this chapter, we will review some existing work related to this problem.

## 3.1 QoS Routing with Inaccurate Information — Theory and Algorithms

In [30], Guerin and Orda investigated the impact of inaccuracies in the available network state and metric information on the path selection process. Faced with inaccurate information, they proposed to select paths by computing path success probabilities. Specifically, they considered two cases: (a) routing with

bandwidth guarantees and (b) with end–to–end delay guarantees.

For the case (a), a probabilistic model is used to capture the inaccuracy of the link states. As a result, each link $l$ is assigned with a $p_l(x)$ value, which is the probability that link $l$ can accommodate a connection with bandwidth requirement $x$. Then the problem of QoS routing is described as follows: find a path $\mathbf{p}^*$ such that, for any path $\mathbf{p}$,

$$\prod_{l \in \mathbf{p}^*} p_l(w) \geq \prod_{l \in \mathbf{p}} p_l(w),$$

where $w$ is the requested bandwidth. This problem can be transformed into the standard *shortest path algorithm* with metric $(-log p_l(w))$ and solved.

The case (b) is broken further into two sub–cases, the *delay–based* model and the *rate–based* model. In the delay–based model, each node advertises the local delay bounds it can guarantee and conducts path computation based on the delay metrics. But the disadvantage is that the local delay is dependent on traffic characteristics, which varies from connection to connection, and therefore the advertised delay tends to be inaccurate. In the rate–based model [16, 67], delay guarantees are mapped into rate guarantees, and nodes advertise the residual rate they have available. Since the residual rate does not depend on the characteristics of new connections, it is a more accurate metric.

In the context of the rate–based model, the end–to–end delay guarantee $d(\mathbf{p})$ can be determined as follows:

$$d(\mathbf{p}) = \frac{\sigma}{r} + \frac{\sum_{l \in \mathbf{p}} c_l}{r} + \sum_{l \in \mathbf{p}} d_l, \tag{3.1}$$

where $\sigma$ is the size of the connection's burst, $c_l$ is the maximum packet length for the connection, and $r$ is the minimal rate for the connection along the path. In this model, the QoS routing problem with delay constraints can be described as follows: given a maximum delay requirement $D$ for a connection, and $\pi_D(\mathbf{p})$, which is the probability that $d(\mathbf{p}) \leq D$ for a path $\mathbf{p}$, find a path $\mathbf{p}^*$ such that, for any path $\mathbf{p}$,

$$\pi_D(\mathbf{p}^*) \leq \pi_D(\mathbf{p}),$$

They proved that this problem is NP–complete and showed that there exists a number of special cases of particular practical significance, for which they proposed tractable solutions.

In the context of the delay–based model, the QoS routing problem with delay constraints $D$ can be described as follows: given a graph $G(V, E)$, the p.d.f.'s $p_l(d)$ for all link $l \in E$, which is the probability that link $l$ will introduce a delay of at most $d$, (i.e., $d_l < d$,) and $\pi_D(\mathbf{p})$, which is the probability that $\sum_{l \in \mathbf{p}} d_l \leq D$ for a path $\mathbf{p}$, the problem is to find a path $\mathbf{p}^*$ such that, for any path $\mathbf{p}$,

$$\pi_D(\mathbf{p}^*) \leq \pi_D(\mathbf{p}).$$

They showed that this problem is also NP–complete and proposed tractable solutions for several special cases of practical importance. [30] laid a theoretical ground for inaccuracy tolerant QoS routing with bandwidth and delay metrics, but performance evaluation is expected to examine the effectiveness of the particular algorithms.

## 3.2 Safety–Based Routing

Based on the idea of using probabilistic model to describe inaccuracy, as described in the aforementioned case (a), In [5], a "safety–based" algorithm was presented with an attempt to compensate for the inaccuracy in the link state information. Instead of using residual/remaining bandwidth as the path computation and selection metric, as commonly used in many algorithms [47, 66], a "safety" value was used. Given the requested amount of bandwidth, the last advertised value of available bandwidth, and the triggering policy parameters, e.g., update triggering threshold, it is possible to compute the range of values for the actual available bandwidth on a link. Based on this range and an assumption about the distribution of the real bandwidth within the range, the probability that the requested bandwidth is available on this link can be obtained. The probability, or the "safety" value, of the path, then can be used to rank links/paths and consequently used in routing computation and selection. By doing so, the scheme avoids dependence on the out–of–date bandwidth

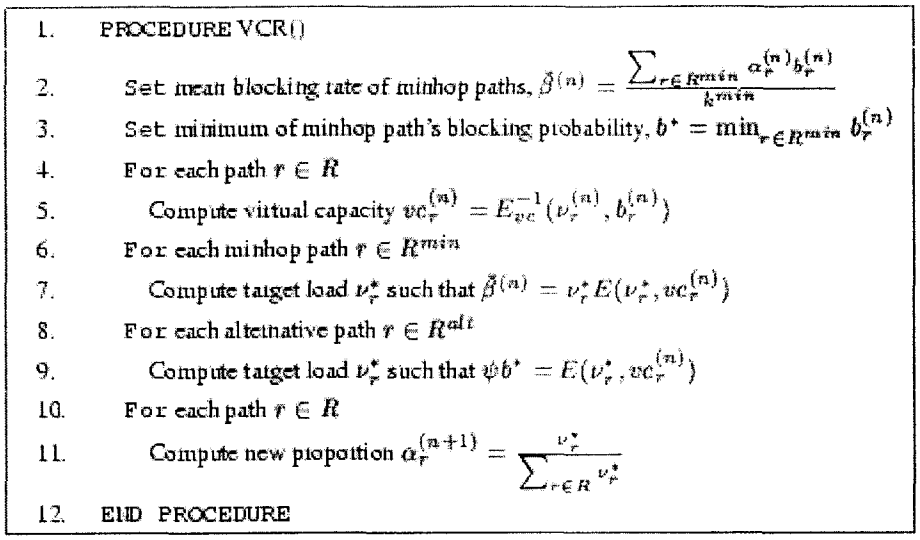


Figure 3.1: Flow proportion computation based on VCR

information and takes inaccuracy explicitly into consideration when making routing decisions.

## 3.3 Localized Routing

In [56], Nelakuditi, Zhang and Tsang proposed a *Proportional Sticky Routing (PSR)* scheme, which does not depend on the global exchange of routing information but requires the local observed blocking information instead. The *PSR* assumed that a number of paths have been pre–computed between source and destination pairs, and it operates in two stages: the *proportional flow routing* stage and the *computation of flow proportions* stage. The proportional flow routing stage is an *observation period.* During this period, a source allocates to each route a certain proportion of traffic, which is calculated in the computation of flow proportions stage. At the end of the observation period, the source computes the blocking probability of each route, based on which the traffic proportion assigned for each route is re–computed with the purpose of balancing blocking rates among all the paths and giving the shortest path higher priority than the alternative paths (i.e., trunk reservation).

Figure 3.1 describes the procedure of a typical flow proportion computation stage using *Virtual Circuit based Routing (VCR).* The procedure is as follows:

1. given a set of pre-computed paths, $R$, and the observed blocking rate for each path, $b_r^{(n)}$, compute the virtual capacity of each path, $vc_r^{(n)}$ using *Erlang's Loss Formula* (line 4 and 5).

2. based on the trunk reservation principle and the objective of equalizing the blocking probability of each path in both the minimum hop paths $R^{min}$ and the alternative paths $R^{alt}$, compute the target blocking rate for $R^{min}$, i.e., $\beta^{(n)}$, and the blocking probability for $R^{alt}$, i.e., $b^*$ (line 1 and 2).

3. with the target blocking rate/probability and the virtual capacity of each path, compute the new target loads for each path, $\nu_r^*$, based on Erlang's Loss formula (line 6 – 9).

4. given the new target loads for all the paths, compute the new proportion of flows, $\alpha_r^{(n+1)}$, for each path $r$, resulting a new load $\nu_r^{(n+1)} = \alpha_r^{(n+1)}$ (line 10 and 11).

Despite the theoretical feasibility of the localized routing scheme, its performance curve is not exciting according to the comparative study conducted in [74] — after all the routing decisions are based on merely the local information which is very limited and not very accurate.

## 3.4 Parameter–Tuning–Based Routing

In [66] the routing performance and implementation overhead were evaluated with respect to a number of parameters: link state update policies, link cost metrics, network topology, and traffic patterns, etc. For example, the influence of different update mechanisms on routing blocking performance and its two composing elements, signaling failure and routing failure, were evaluated. By observing that signaling failure is more expensive in that it consumes processing resources and delays the establishment of other connections, it was suggested to use a hybrid update policy with a moderately large update trigger value and a small hold–down timer. Subsequently, the impact of the above pa-

rameters on the routing performance and implementation overhead was studied. By carefully tuning the parameters, the results in [66] suggested that a balance can be achieved between high accuracy and low complexity/better performance. The limitation of the work is that it is based on cooperation of existing techniques, which provides limited space to increase performance. Aside from that, the study in [66] is based only on one kind of topology — the regular $k$–array $n$–cube topologies.

## 3.5 Multi–Path Routing: Ticket–Based Routing

In [63], Chen and Nahrstedt proposed a distributed scheme, called *ticket–based probing*, which searches multiple paths in parallel to find a delay–constrained least–cost path. In particular, every time a connection request arrives, a source sends out some probes, each of which is composed of a number of tickets. These tickets collect the delay and cost information along the paths until arriving at the destination. At an intermediate node $i$, if a ticket probes that the delay it has experienced so far plus the estimated delay from $i$ to the destination violates the delay bound, then the ticket will be dropped and won't be able to reach the destination. Those tickets which successfully make it to the destination indicate successful paths, from which the one with the least cost is selected. The above procedure requires each node to estimate the delay from itself to the destination, and this is done based on their delay estimation model:
$D_i(t)$ keeps the minimum end–to–end delay from $i$ to $t$.
$\Delta D_i(t)$ keeps the estimated maximum change of $D_i(t)$ before the next update.
$D_i^{old}(t)$ and $D_i^{new}(t)$ are the values of $D_i(t)$ before and after the update.
$\Delta D_i^{old}(t)$ and $\Delta D_i^{new}(t)$ are the values of $\Delta D_i(t)$ before and after the update.
$\Delta D_i^{new}(t)$ is calculated as:
$\Delta D_i^{new}(t) = \alpha \times \Delta D_i^{old}(t) + (1-\alpha) \times |D_i^{new}(t) - D_i^{old}(t)|$,
and the end–to–end delay from $i$ to $t$ is estimated to be between $D_i(t) - \Delta D_i(t)$ and $D_i(t) + \Delta D_i(t)$ in the next update period.
Their scheme is an effective heuristic to find delay–constrained least–cost

paths, but since a number of probes are sent out the network upon each connection request, the routing information overhead is large.

## 3.6 Multi–Path Routing Combined with Resource Reservation

Cidon et al. [21] also made use of multiple paths to make reservation along several paths in parallel. They proposed four reservation schemes: *the fast algorithm, the logarithmic algorithm, the superfast algorithm, and the slow algorithm.* The first three are flooding–based, which attempt to establish a connection fairly fast at the cost of path optimality: a *Request* message originates from a source and are duplicated on their way to the destination as long as there is resource available on the way; if intermediate nodes do not have enough resource, then *Reject* messages are issued to *early release* the reserved bandwidth; the destination issues an *Accept* message upon receiving the first *Request* message. In contrast, the slow algorithm pursues optimality of a reservation path at a price of reservation speed: a node waits until it receives messages from all its incoming links before it sends *Reject* messages to upstream links. In summary, the flooding based multi–path algorithms realize fast reservations but sacrifice path optimality. Also it is not a scalable solution because of its high message complexity, especially considering that reservation is made on a per–connection base. The slow algorithm obtains optimal paths at a price of slow reservation. Put into a per–flow reservation context, this is also not a scalable solution.

## 3.7 Summary

In [74], Yuan, Zheng and Ding conducted a comparative study of some of the previously mentioned QoS routing schemes: safety–based routing, randomized routing, ticket–based multi–path routing, and localized routing. They divide imprecision into two categories: *random imprecision*, caused by non–negligible propagation delay or infrequent link state updates, and *deterministic imprecision*, caused by threshold–based or class–based link state update policies.

Their performance study shows that randomized routing, which randomly chooses a path from a set of feasible paths computed based on the imprecise global network state information, performs the worst. The ticket–based multi–path routing performs the best, being able to deal with deterministic imprecision as well as random one. In contrast, the safety–based routing is effective in dealing with deterministic imprecision but ineffective in dealing with random imprecision. The localized routing offers much worse performance than the ticket–based multi–path routing scheme. Their study shows that the only class of schemes with a promise of widely acceptable performance appears to be multi–path routing. Therefore, in the next Chapter, we will present our multi–path QoS routing solution.

# Chapter 4

# Multi–Path Routing in Networks with Inaccurate Information

Our work is based on the IntServ/RSVP model, where each connection needs to make a reservation in order to receive guaranteed routing services. Similar to [21] and [63], multiple paths are used when making reservations, but unlike their approaches, which construct reservation paths upon the arrival of each connection request, our approach pre–computes multiple paths, along which per–flow reservations are to be made. The purpose of the pre–computation is to reduce the cost of path construction and routing information exchange; also, making reservations along several pre–established paths of limited length causes lower reservation signaling overhead than the flooding–based reservation mechanism used in [21, 63]. On the other hand, path pre–computation reduces the accuracy of the paths; therefore we enlarge the success opportunity of reservation by sequentially scanning all the multiple paths until success instead of in parallel searching for only one path, as in [21, 63].

In our study, the QoS metrics of concern are the *bottleneck bandwidth* (defined below) and the number of hops. As pointed out in [75] and [49], certain QoS metrics (notably, bandwidth, delay and delay jitter) are not independent when specific scheduling policies are used. In particular, with WFQ–like scheduling, the end–to–end delay and delay jitter depend on the requested bandwidth [75] and can be translated into bandwidth requirements [61]. Con-

sequently, several studies, e.g., [66], consider bandwidth as the sole metric in route computation. As stated in [30], this simplifies to some extent the path computation process, so that tractable solutions can be provided in a number of interesting cases. Following this approach, we do not take delay metrics into consideration in this study. The second metric of our concern is the number of hops, a good indicator of the total amount of resources used along the path and an important factor from the viewpoint of economy, efficiency and global performance of the network. Our studies indicate that regardless of the other criteria, it always makes sense to keep the connection path as short as possible, as long as it fullfills the QoS constraints.

We develop two categories of $K$–shortest routing algorithms, the *hop–based* and the *bandwidth–based*, and correspondingly five path selection algorithms: *Best–K–Widest* (BKW), *Random–K–Widest* (RKW), *Shortest–K–Widest* (SKW), *Best–K–Shortest* (BKS), and *Widest–K–Shortest* (WKS). In order to appreciate the impact of the plurality of paths ($K > 1$), we extensively investigate the performance of the proposed routing scheme in large networks (up to 1000 nodes) and topologies that have been [50] demonstrated to capture the current Internet topology in a realistic way.

Our results demonstrate that routing based on multiple paths gives better performance in terms of blocking rates and load balancing features than single–path solutions, if the link state information is inaccurate. We also investigated the cost/performance of our scheme with inaccurate information as compared to the single–path scheme with accurate information. Furthermore, we find that the hop–based algorithms, i.e., ones using hop count as the primary metrics, tend to outperform the bandwidth–based solutions, and that the network topology has a paramount impact on the behavior of a routing algorithm. Last but not the least, we show that some multiple path solutions may fail to take advantage of their supposedly "bigger choice," if they do not account properly for the possible inaccuracies in link state information.

## 4.1 The Routing Model

Our routing model is a link state–based one. Link state routing requires each router to maintain a link state database, which is essentially a map of the network topology with associated resource allocation. Each node periodically exchanges its own link state information with all the other nodes on the network using flooding. The *link state update period* (LSUP) reflects the accuracy of the routing information: the larger the LSUP is, the less accurate the information is. Obviously, there is a trade–off between the accuracy of the link state database and the cost of performing those updates both in terms of the extra network bandwidth and the processing time at the routers. We assume in our model that after every update, each node immediately has the full knowledge of the current link states in the whole network. This is justifiable because the propagation delay of a LSA message is generally small compared to the length of the update period and to the duration of a traffic session.

After the link state database at each router is updated, each router re–calculates its routing tables to all the destinations. Therefore, routes are precomputed periodically, with the computation period equal to the LSUP. At the time of path computation, the router calculates top $K$ best paths from itself to all the other nodes.

Every time a connection arrives, reservation starts along the path which is selected from the top $K$ paths using a certain path selection algorithm. The actual ordering of those paths depends on the specific scheme being used and will be detailed in subsequent sections. If the connection cannot be established via one of the paths, e.g., due to the insufficient remaining bottleneck bandwidth, the router picks one of the remaining paths and tries again. The request is blocked if no path is found after all $K$ of them have been attempted.

## 4.2 Path Construction

Our path construction algorithm [38] is a label–setting algorithm based on the Optimality Principle and being a generalization of Dijkstra's algorithm [52],

which we have modified to find $K$ *one-to-all* loopless paths instead of *one-to-one* non-loopless paths. Its space complexity is $O(Km)$, where $K$ is the number of paths and $m$ is the number of edges. By using a pertinent data structure, its time complexity can be kept at the same level $O(Km)$ [52].

With the hop count and path bottleneck bandwidth used as two separate metrics, our algorithm generates $K$ paths that are either *shortest* in terms of the number of hops (*hop-based algorithms*), or *widest* in terms of the bottleneck bandwidth (*bandwidth-based algorithms*). In order to limit the impact of requests for trivially small bandwidth, a threshold is used to prune unsuitable links right away at the path construction stage. Formally, the multi-path construction problem to be addressed in this section is described as follows:

*Given a network represented by a Directed Acyclic Graph (DAG) $G(V, A)$, the capacity of each link, $b_l$, where $l \in A$, a source node $s$, and a bandwidth threshold $B$, find the best $K$ paths from $s$ to all the other nodes, $\{P_t^k | 0 < k \leq K, t \in V, t \neq s\}$, such that $b(P_t^k) \geq B$, $\forall 0 < k \leq K$, $t \in V, t \neq s$, where $b(P_t^k) = min_{l \in P_t^k} b_l$ is called the bottleneck bandwidth of path $P_t^k$.*

The path construction algorithm is described as follows:

Let a DAG $(V, A)$ denote a network with $n$ nodes and $m$ edges, where $V = \{1, \ldots, n\}$, and $A = \{a_{ij} | i, j \in V\}$. The problem is to find the top $K$ paths from source $s$ to all the other nodes. Define a label set $X$ and an one-to-many projection $h : V \rightarrow X$, meaning that each node $i \in V$ corresponds to a set of labels $h(i)$, each element of which represents a path from $s$ to $i$. Each label/path is associated with a major weight and a minor weight. For the hop-based algorithm, the major weight is the inverse of the number of hops and the minor weight is the bottleneck bandwidth of the path represented by this label. Those weights are interchanged for the bandwidth-based algorithm. The minor weight is not used by the path construction algorithm (except for being computed), but it is needed later for path selection. We introduce the following notation:

| | |
|---|---|
| $s$ | the source node |
| $X$ | the label set |

| | |
|---|---|
| $b_{vj}$ | the link bandwidth from $v$ to $j$ |
| $count[v]$ | the number of paths found so far from $s$ to $v$ |
| $lb0$ | the *permanent* label selected from $X$ (such that $lb0.bw \geq lb.bw, \forall\ lb \in X$) |
| $lb0.ver$ | the node corresponding to label $lb0$ |
| $lb0.bw$ | the bottleneck bandwidth of the path from $s$ to $lb0.ver$ |
| $lb0.parent$ | the label that generated $lb0$ |
| $P_v(count[v])$ | the $count[v]$-th path from $s$ to $v$ |
| $lb1$ | a new label generated from $lb0$ |

The following algorithm calculates the $K$ best paths from source $s$ to all destinations according to the major weight:

$count[i] = 0, \forall\ i \in V$
$lb0 = 1$
$lb0.ver = s$
$lb0.bw = infinite$
$lb0.hops = 0$
$lb0.parent = NULL$
$X = \{lb0\}$
**while** ( $X \neq \emptyset$ and $\exists\ i$ such that $count[i] < K$,
    where $0 < i < n$)
**do** begin
  find a permanent label $lb0$ from $X$, such that
    $lb0.bw \geq lb.bw, \forall\ lb \in X$
  $X = X - \{lb0\}$
  $v = lb0.ver$
  $count[v] = count[v] + 1$;
  **if** ($count[v] \leq K$ and $lb0.hops < Max\_Hop\_Num$)
  **then** begin
    record the path $P_v(count[v])$ by
      following the $lb0 \rightarrow parent$ link
    **for** each $a_{vj} \in A$  /*generate new labels*/
    **do** begin
      **if** (the potential new label does not result in
          a loop and $b_{vj} > bw\_thrsh$
      **do** begin  /*generate this label*/
          $lb1.ver = j$
          $lb1 = lb0 + 1$

```
                lb1.bw = min(lb0.bw, b_vj)
                lb1.hops = lb0.hops + 1
                lb1.parent = lb0;
                X = X ∪ {lb1}   /*add it into the label set*/
            end
        end
    end
end
```

## Proof

Without loss of generality let us consider the hop based algorithm, and let the $K$ labels found for node $v$ be $lbl_i, \forall\ 1 \leq i \leq K$. Suppose that if we continue executing the algorithm, we will find a new path from $s$ to $v$, with a hop number smaller than one of the $K$ determined ones. That is, the algorithm will generate a new label $lbl_0$, such that $\exists\ 1 \leq i \leq K$, $lbl_0.hops < lbl_i.hops$. Naturally we get $lbl_0.parent.hops < lbl_i.parent.hops$ (because $l.parent.hops = l.hops + 1, \forall\ l \in X$). This implies that the $lbl_0.parent$ must have been marked as a permanent label earlier than the $lbl_i.parent$, because in each step of the algorithm, only the best label is marked as a permanent label. If this is true, it should be $lbl_0$, not $lbl_i$, that was generated earlier—a contradiction. The bandwidth based case can be proved in a similar way.

# 4.3 Path Selection

Below we list five path selection algorithms resulting from two different major criteria and different ways of applying the minor criteria.

BKW *Best-K-Widest*: from the $K$ widest paths, select the one whose bottleneck bandwidth most tightly fits the request bandwidth (best-fit).

RKW *Random-K-Widest*: from the $K$ widest paths, select one at random.

SKW *Shortest-K-Widest*: from the $K$ widest paths, select the one with the least number of hops.

BKS *Best–K–Shortest*: from the $K$ shortest paths, select the one whose bandwidth most tightly fits the connection request.

WKS *Widest–K–Shortest*: from the $K$ shortest paths, select the one with the largest bandwidth.

Note that, although the names *Shortest–K–Widest* and *Widest–K–Shortest* resemble *Shortest–Widest* from [70] and *Widest–Shortest* (WS) from [3], our algorithms are quite different from those previously proposed solutions. In particular, our SKW finds the top $K$ paths, while the *Shortest–Widest* algorithm of [70] uses the "shortest–widest" constraint during the (single) path construction phase. That is, when multiple labels with the same bandwidth are found, only the shortest one is permanently labeled and the others are deleted. SKW maintains all the widest labels for later use and is therefore able to find top $K$ paths. It decides on the "shortest" path during the path selection phase rather than when the path is being computed.

Furthermore, note that although the *widest–shortest* algorithm from [3] also generates multiple paths for a given destination, it is different from WKS in that it provides fewer choices for selecting short paths, and the resulting connection is thus likely to need more resources. In particular, it uses an upper bound for the hop count and for each hop count, it only keeps one widest path. Consequently, it ignores the equal–hop–count multiple paths. Besides, the *widest–shortest* algorithm requires that the $(i+1)$–th path be "wider" than the $i$–th path. This makes sense when the link state information is accurate, but if it is not the case, some feasible paths may be incorrectly ignored. Thus, the very nature of the algorithm in [3] impairs its performance when the link state information is inaccurate. We will make a comparison between this scheme and our proposed one in Section 4.5.

## 4.4 The Simulation Model

### 4.4.1 Performance Metrics

We are primarily interested in three performance measures as functions of the offered load and link state update period: the connection bandwidth blocking rate $\theta$, the *coefficient of variation* (Cov) of the link utilization and the protocol overhead. The blocking rate is weighted by the connection bandwidth [47, 49] and defined as:

$$\theta = \frac{\sum bandwidth\_of\_blocked\_connections}{\sum bandwidth\_of\_requests}$$

The *coefficient of variation* of the link utilization captures the variability of the load between different links and indicates how well the offered load is spread over the network resources. We define it as:

$$Cov(LU) = \frac{\sigma(LU)}{\mu(LU)},$$

where $LU$ stands for link utilization (i.e., the fraction of time the link is being used), $\sigma$ and $\mu$ are the Standard Deviation and mean respectively.

The protocol overhead in our routing protocol consists of the signaling overhead and the link state exchange overhead. With QoS routing, we need to make a reservation along the selected path before routing a request. The cost expended during this procedure is called signaling overhead, represented by the number of hops experienced by signaling messages. The link state exchange overhead is represented by the number of hops experienced by link state update messages. As demonstrated in the simulation, the link state update overhead is the dominating overhead.

### 4.4.2 Topologies

We consider four different network topologies: two artificial ones, i.e. the power–law based topologies [25] and the regular torus ones, and two real–life ones, the MCI [47] and Cluster [47] topologies.

The most representative of our simulation results have been obtained for random network configurations built by the generator of Magoni and Pansiot [50]. Reasonably large networks generated by this program have been

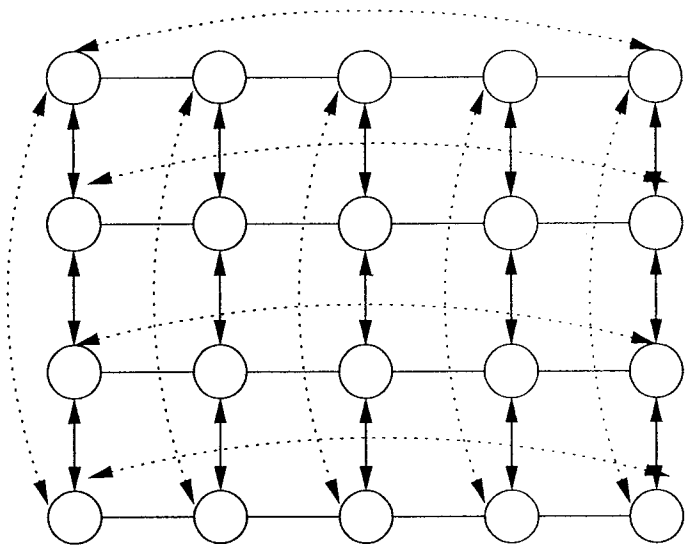

Figure 4.1: A 4X5 torus

demonstrated to obey the most relevant (from the viewpoint of routing) power laws found in existing wide area networks (e.g., sections of the Internet). A random network configuration in our experiments is characterized by two parameters: the number of nodes, $N$, and the average node degree, $\delta$. A typical value of $\delta$ found in the Internet is 2.5 [50]. Besides, we consider sparser networks, with $\delta = 2.0$, as well as denser networks, with $\delta = 2.9$. For reliability, a single experiment has been verified on a number of different topology samples generated for a given pair $< N, \delta >$. For a network of a reasonably large size ($\geq$ 500nodes), the obtained results are highly consistent across different topology samples (obeying the same power laws).

For reference, we also consider regular torus networks with the same populations of nodes as the power law configurations. Figure 4.1 illustrates an example torus graph. Certain performance measures, notably our *coefficient of variation* may be affected by the inherent irregularities present in any random network, even if it is large and obeys reasonable statistical laws. A regular topology eliminates those problems and helps us find out how much our performance measures are influenced by statistical aberrations in network configurations. Every link in our network has the same bandwidth of 155Mbps unless otherwise specified.

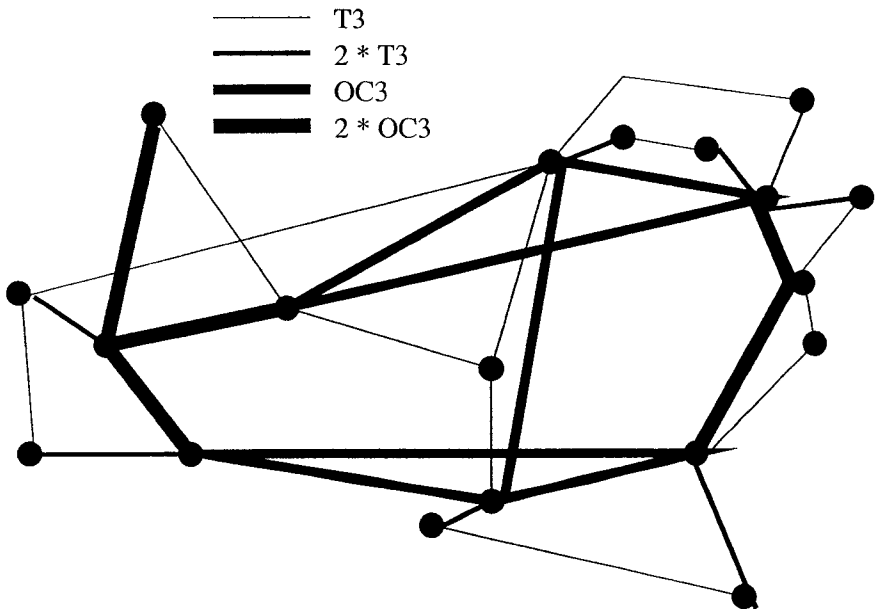


Figure 4.2: The MCI topology

Last but not the least, we also investigated the performance of the proposed schemes in real–life networks, i.e., the 19–node MCI topology [47] (see Figure 4.2) and the 16–node Cluster topologies [47].

### 4.4.3 Traffic

The offered traffic consists of randomly generated sessions with exponentially distributed interarrival time. The mean value of this distribution is a simulation parameter that determines the global load in the network. Following [12], the connection holding time is sampled from a log-normal distribution with the mean of 3 minutes. The requested bandwidth of a connection is uniformly distributed between 1 and 5 Mbps. Additionally, the arriving traffic can be evenly distributed over the entire network, i.e., uniformly selecting the source–destination pair from the network, or biased, i.e., favoring some source–destination pairs.

The link state information is updated periodically, with LSUP values varying from 0 to 100 minutes, creating scenarios with different levels of accuracy.

The total amount of simulated time for a single experiment was originally set at 24 hours, and six independent experiments were used to obtain a single

point of a performance curve. The high observed consistency of results allowed us to reduce the amount of simulated time by half, and the number of samples by the same factor.

## 4.5 Results

The goal of our experiments is to answer the following questions:

1. Is there a path selection algorithm (among the ones listed in Section 4.3) that gives consistently the best performance?

2. How does the multiple–path approach fare in the context of inaccurate link state information? In particular, how does the performance of our multiple–path routing scheme depend on the LSUP?

3. How large should the multiple–path selection be, i.e., how does $K$ (the number of paths to choose from) affect the performance of our routing scheme?

4. What are the costs and benefits of the proposed multiple–path algorithms?

5. How does our routing scheme compare to other schemes?

### 4.5.1 Comparison between the *Hop–Based* and the *Bandwidth–Based* Algorithms

Figure 4.3, obtained for the MCI network, illustrates the blocking rates under all five path selection algorithms with the number of alternative paths $K = 3$. The trend shown in this figure has been clearly visible in all our experiments. First, it turns out that the hop–based selection algorithms (i.e., WKS and BKS) unquestionably outperform the bandwidth–based ones (BKW, RKW and SKW), with WKS winning over BKS. Although the superiority of WKS over BKS is not obvious in Figure 4.4, it becomes visible in a larger/denser network and the MCI network , e.g., see Figure 4.5 and Figure 4.3.

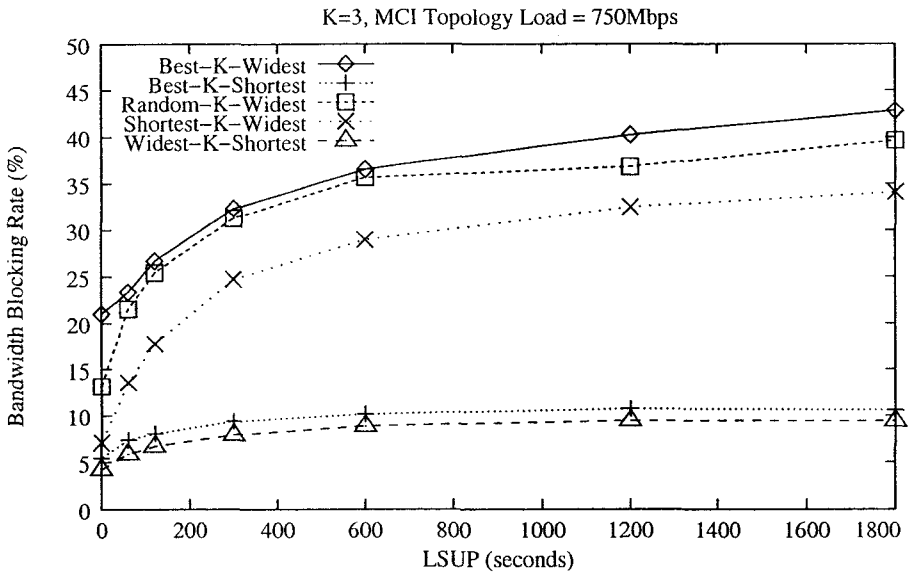


Figure 4.3: The MCI network, $K = 3$

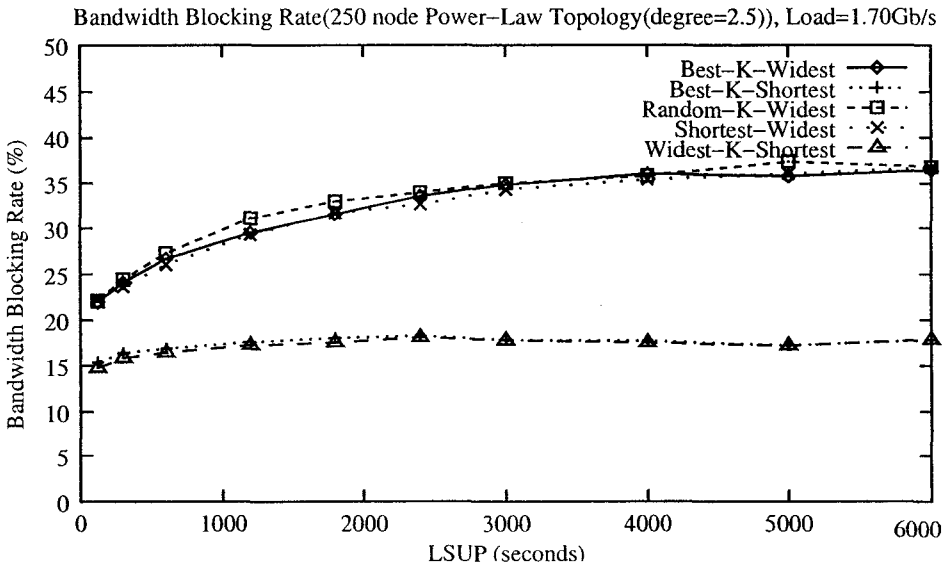


Figure 4.4: Small typical networks, $K = 3$

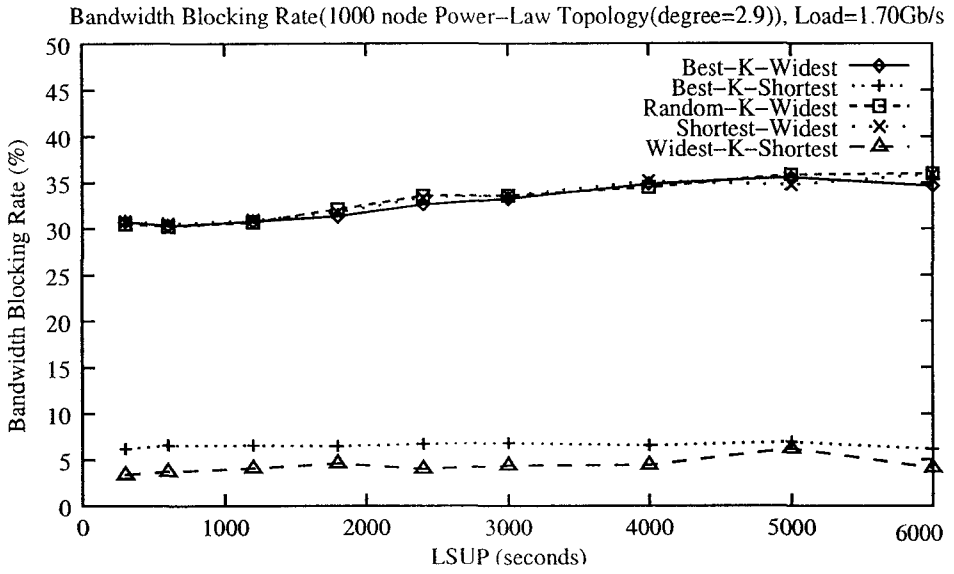


Figure 4.5: Large dense networks, $K = 3$

The reason why the hop–based algorithms win in this competition is that by making the path length the primary optimization criterion they focus on minimizing the amount of resources needed to sustain a connection. Consequently, the network tends to use less of its total bandwidth per session and is thus able to accommodate more connections.

Figures 4.3, 4.4, and 4.5 also demonstrate that past some narrow initial range of the LSUP, the quality of our routing schemes (especially the hop–based ones) is not adversely affected by the inaccuracies in the link state information. In all cases, as the link state update period goes to infinity, the blocking rate stabilizes to a constant. This is not surprising, since then the calculated paths are not much better than ones selected at random. A more important observation is the relative insensitivity of the hop–based schemes to the link state update period.

Aside from blocking performance, we also made a comparison between the link state error distribution associated with the two types of algorithms. Based on Figure 4.6 and using a *quantile–quantile* plot method [37], we found that the error is approximately normally distributed. The figure indicates that the hop–based algorithms generate a smaller error variance . According to the graph, we found that with the hop–based algorithm, BKS, 98.40% errors lie in

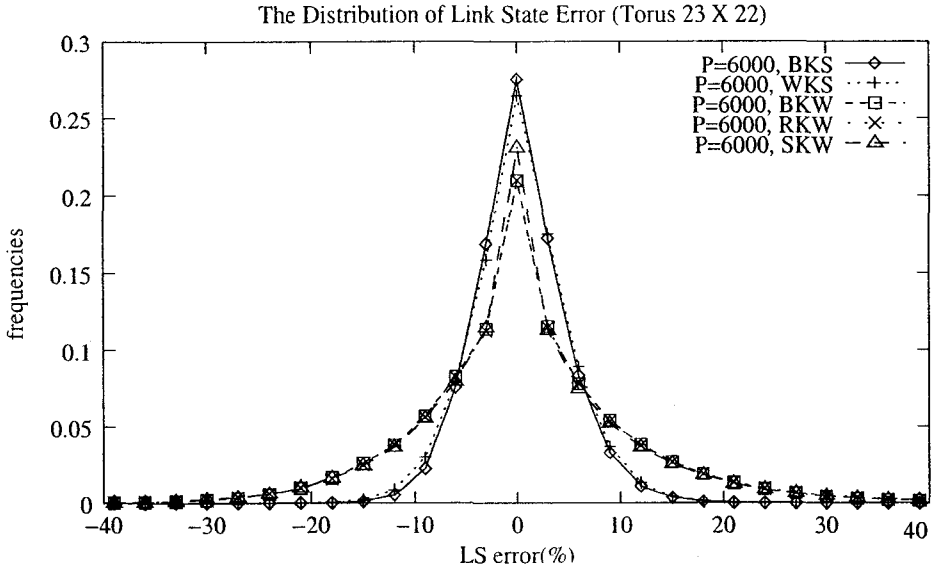


Figure 4.6: Link state error distribution

a range as narrow as [−10%, 10%], while with the bandwidth–based algorithm, BKW, only 66.41% of errors are in that range, and the remaining 35.59% are out of this range, meaning that the hop–based algorithm generates smaller link state errors than the bandwidth–based one.

### 4.5.2 Impacts of the Number of Alternative Paths

To answer questions 2 and 3, we examined the impact of the $K$ on the performance of a path selection algorithm, and found it to depend on the algorithm and on the network density. Fortunately, the hop–based schemes perform quite well with a small number of alternative paths to select from. Figure 4.7 and 4.8 show the blocking rates of WKS in two medium–sized networks for different values of $K$. In the sparse network, the impact of $K$ is quite negligible, which means that alternative path selection brings about little, if any, improvement. This is understandable, because low connectivity implies a small number of alternative paths with comparable length (using a comparable amount of network resources). Consequently, even though multiple alternative paths may still exist, they tend to be of different length, so selecting an alternative to the shortest path in such a sparse network is statistically a poorer choice than in a network offering multiple shortest paths. In Figure 4.8, we see a clear

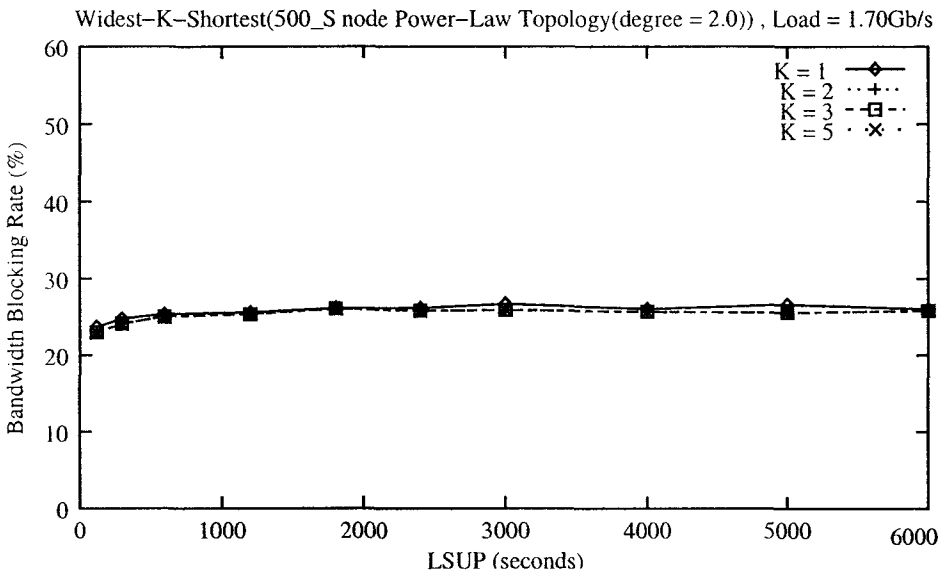


Figure 4.7: Medium sparse networks, different values of $K$

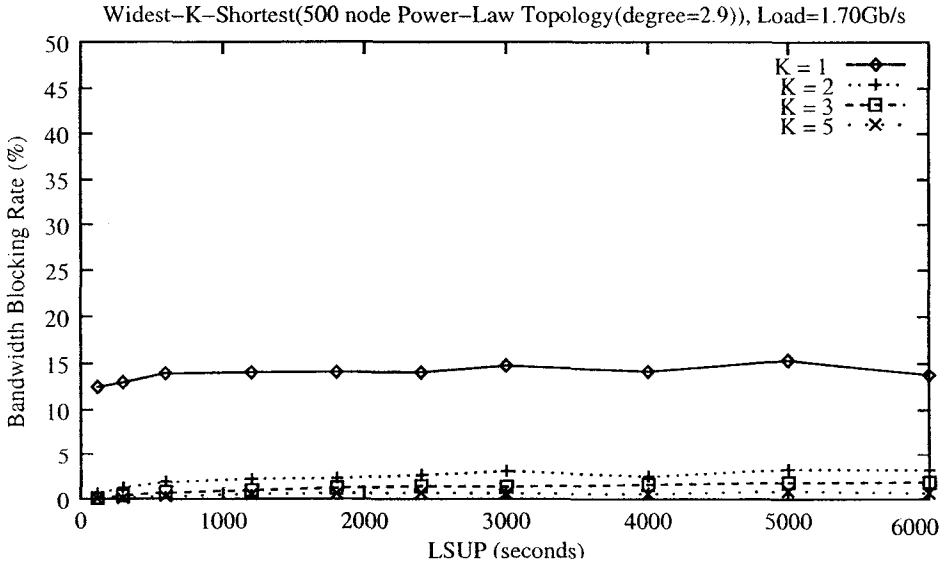


Figure 4.8: Medium dense networks, different values of $K$

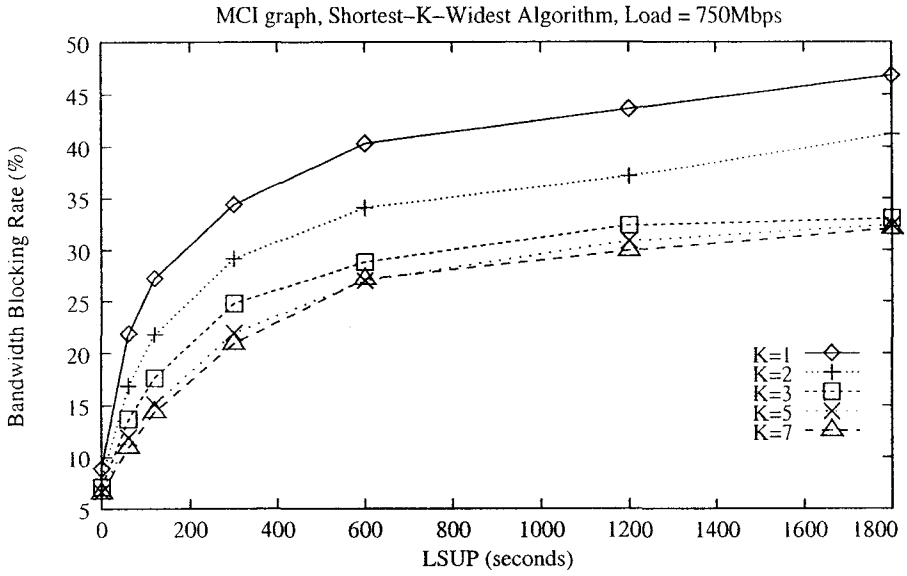


Figure 4.9: The MCI network, different values of $K$

improvement caused by the statistical presence of sensible alternatives. Figure 4.9 demonstrate the blocking performance of the SKW algorithm for 5 increasing $K$ values in the MCI network. Again, it confirms the intuition that a larger $K$ implies a lower blocking rate.

Regardless of the circumstances, the gap between the cases $K = 1$ and $K = 3$ is significantly bigger than between $K = 3$ and $K = \infty$. This indicates that while it is advantageous to have a choice, that choice need not be excessively big. This observation is good news. It means that the complexity of the routing algorithm can be well contained, because all it needs to do is to find just a few (e.g., 3) alternative paths, which effort is only moderately bigger than finding a single path.

### 4.5.3 Load Balancing Features

Our primary intuition behind multiple paths was that with several alternative routes, we would be able to spread the offered load more evenly over the entire network. Thus, one would expect that the better path selection algorithms should result in a lower observed value of the *coefficient of variation* of the link utilization, $Cov$.

Alas, as indicated by Figure 4.10, this kind of study must be performed

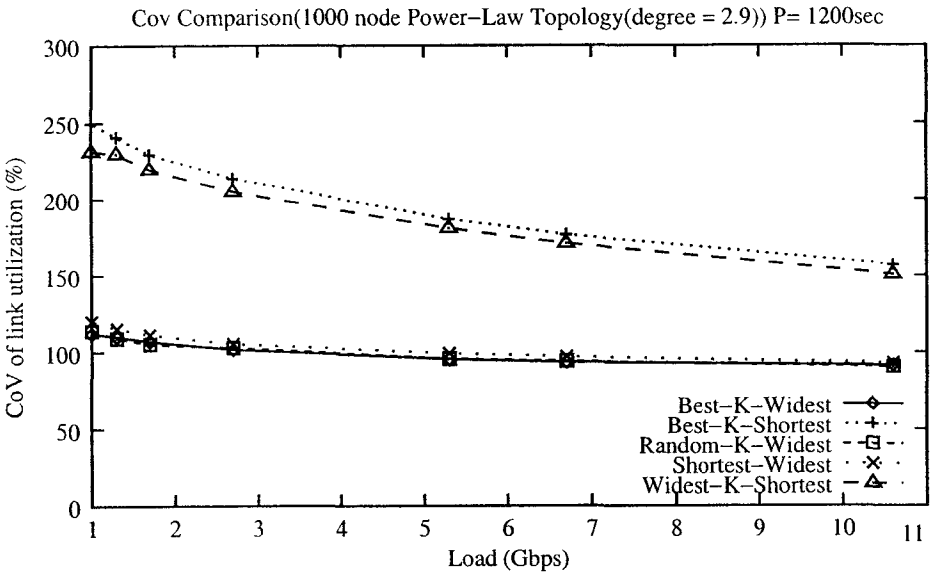


Figure 4.10: Large dense networks, $K = 3$

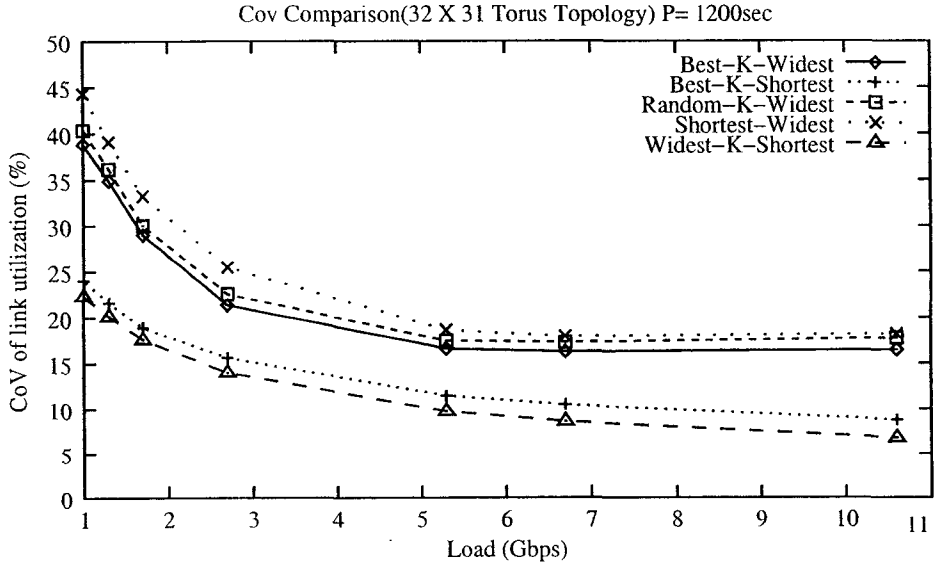


Figure 4.11: Large regular networks, $K = 3$

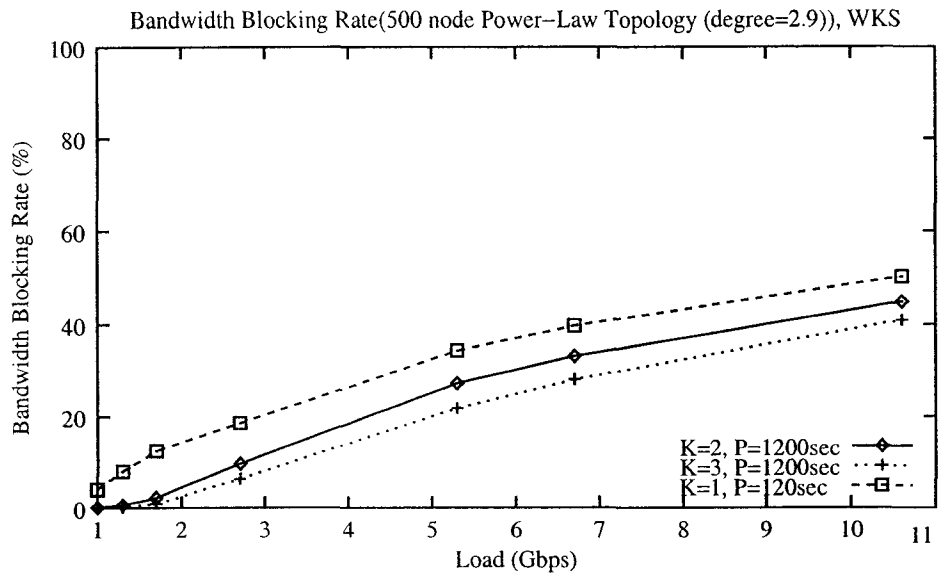


Figure 4.12: Blocking rate comparison of multiple path and single path routing

on a regular topology to make sense. With local irregularities that are bound to occur in a random network of any size, by trying to consistently follow the shortest paths, a routing scheme may quite legitimately create localized hot spots. Those departures from a uniform spread of the offered load will tend to increase with the decreasing average length of a connection path, because the irregularities in topology manifest themselves on a small scale, and they tend to disappear when we look at larger regions of the network. Consequently, in Figure 4.10, the observed value of $Cov$ is higher for the path selection schemes that offer better performance, i.e., target shorter paths. On the other hand, in Figure 4.11, obtained for a perfectly regular torus network, the *coefficient of variation* shows a reversed trend, which remains in a perfect agreement with intuition.

### 4.5.4 Alternatives: Accurate Single Path Routing vs. Inaccurate Multi–Path Routing

So far we have only examined the performance improvement of the proposed multiple–path scheme, in the following we will investigate whether it is a cost effective solution. The comparison is made between two alternatives: WKS with $LSUP = 1200$ and $K = 2$ (or $K = 3$), representing multi–path rout-

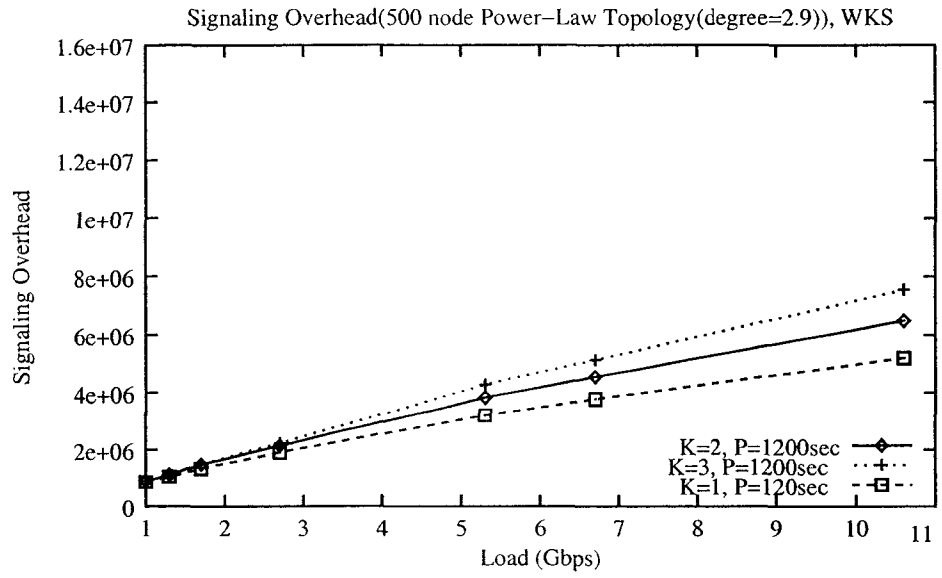


Figure 4.13: Signaling overhead comparison of multiple–path and single path routing

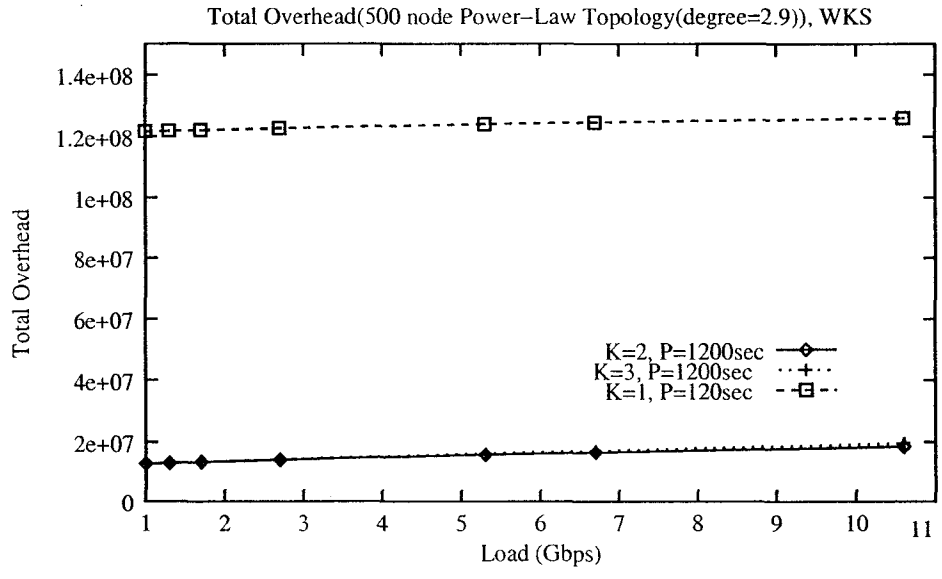


Figure 4.14: Total overhead comparison of multiple–path and single path routing

ing with inaccurate information, and $LSUP = 120$ and $K = 1$, representing single–path routing with accurate information. Figure 4.12 shows that WKS with $LSUP = 1200$ and $K = 2$ or $K = 3$ outperforms the shortest path routing scheme with $LSUP = 120$. In other words, the multiple–path scheme with inaccurate information wins over the single path scheme with accurate information. This performance benefit is obtained at a cost of increased signaling overhead, as shown in Figure 4.13. Nevertheless, as revealed in Figure 4.14, the total cost of the multiple–path scheme with inaccurate information is far less than that of the single path scheme. By total cost, we mean the summation of the signaling overhead and link state exchange overhead. Therefore, we can say that the proposed multiple–path scheme is a scalable solution.

### 4.5.5 Comparisons with Other Schemes

In Section 4.3, we introduced a well–known QoS routing algorithm, WS [3], and compared the difference between the WKS and WS. In [47, 49], Ma introduced and compared several routing algorithms, advocating the *Dynamic–Alternative (DA)* as the best–performing solution under QoS constraints. The confrontation of WKS with the DA and WS illustrates why routing schemes that do not account for the inaccuracies in the link state information may perform poorly, even if they appear superior when that information is accurate. This comparison has been performed on real topologies, i.e., the MCI topology used in [49], regular topologies, such as torus, and power–law topologies. From Figure 4.15, Figure 4.16, Figure 4.17 and Figure 4.18, we observe that if the link state information is accurate ($LSUP = 60sec$), DA and WS exhibit almost the same performance as WKS, if not better. However, they perform worse than WKS with outdated ($LSUP = 1200sec$) link state information. Figure 4.19 and Figure 4.20 show that WKS, even in the case of $K = 2$, wins over DA and WS with either $K = 2$ or $K = 3$.

In Section 4.3, we have analyzed why the WKS is better suitable for inaccurate information than the WS, and in the following, we will explain why WKS is superior to DA in an inaccurate environment. Assume that for a given source–destination pair,

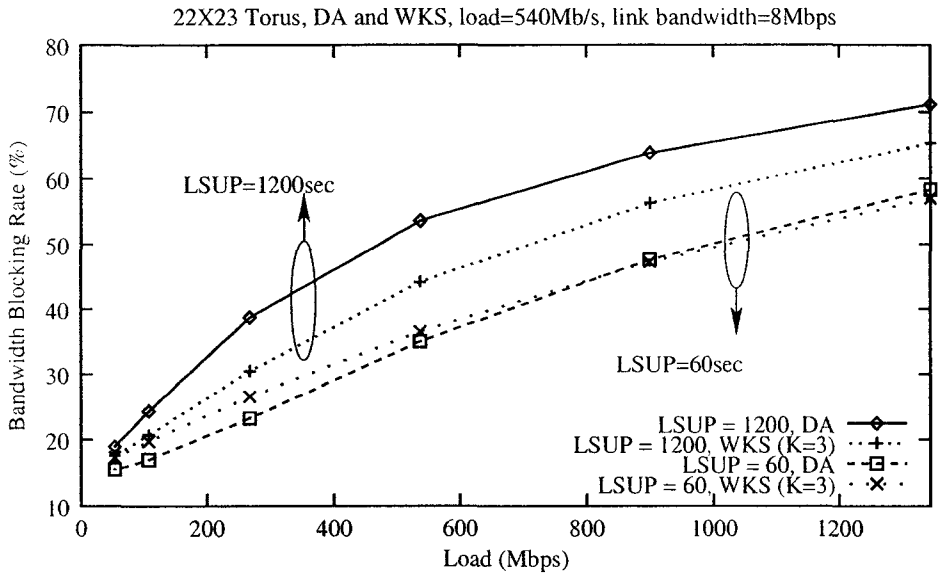


Figure 4.15: 22X23 torus, Blocking rate vs. Load of DA and WKS

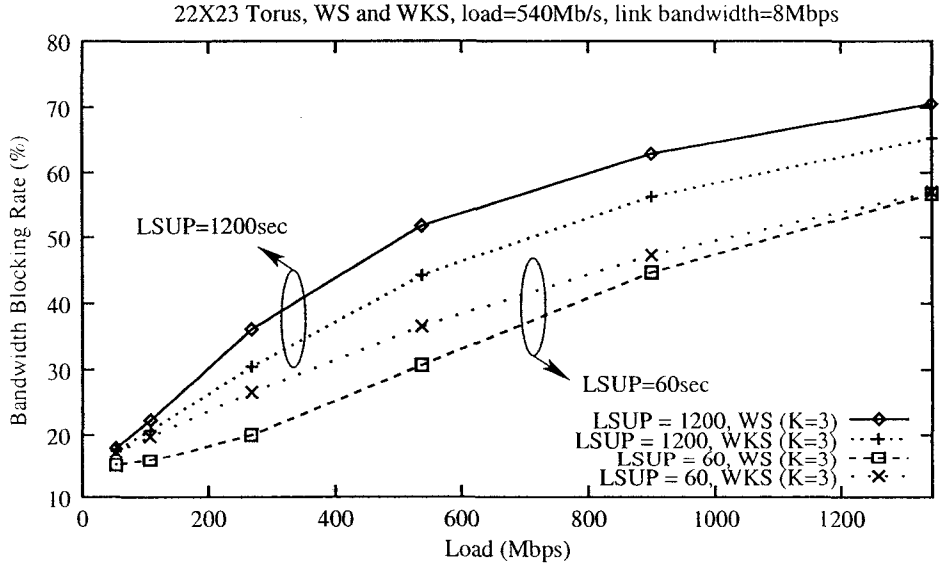


Figure 4.16: 22X23 torus, Blocking rate vs. Load of WS and WKS

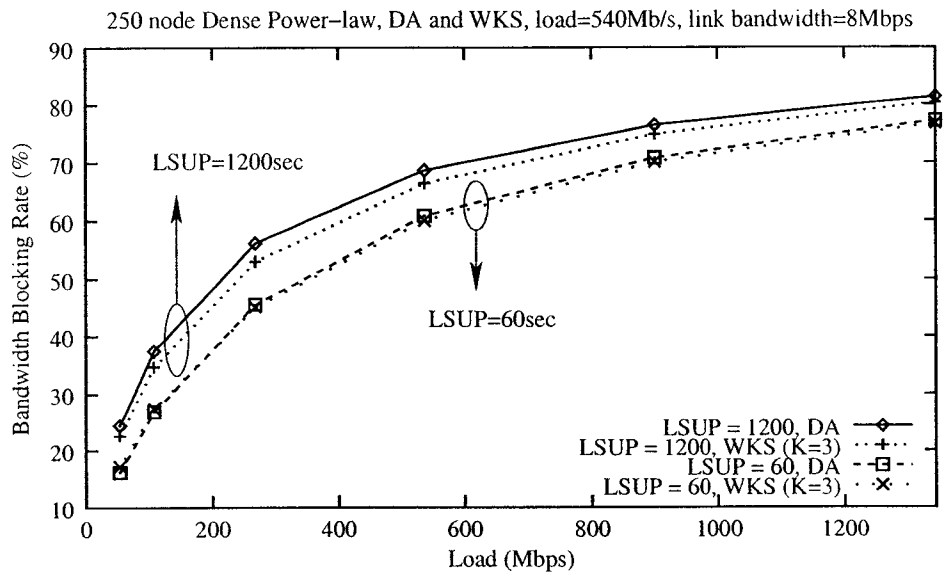


Figure 4.17: 250 node dense power–law, Blocking rate vs. Load of DA and WKS

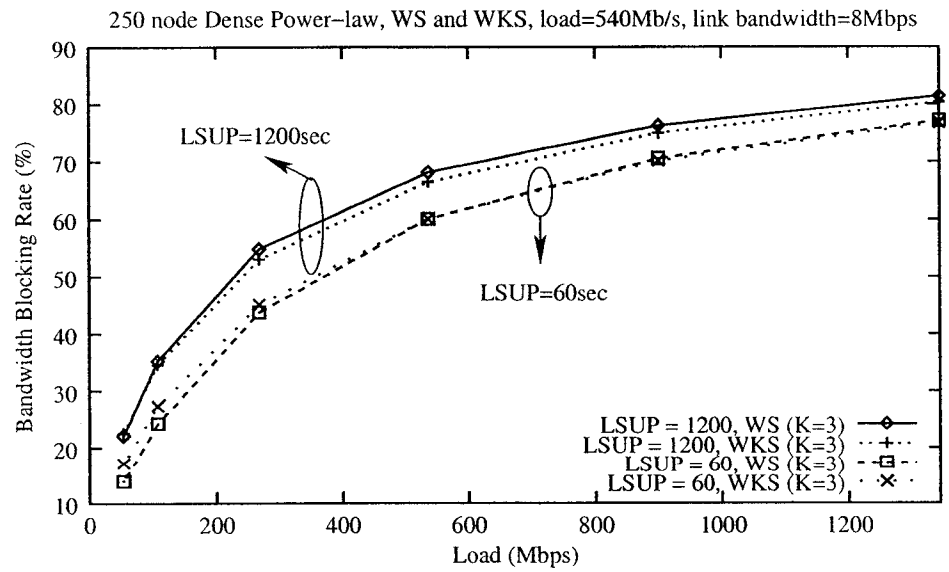


Figure 4.18: 250 node dense power–law, Blocking rate vs. Load of WS and WKS

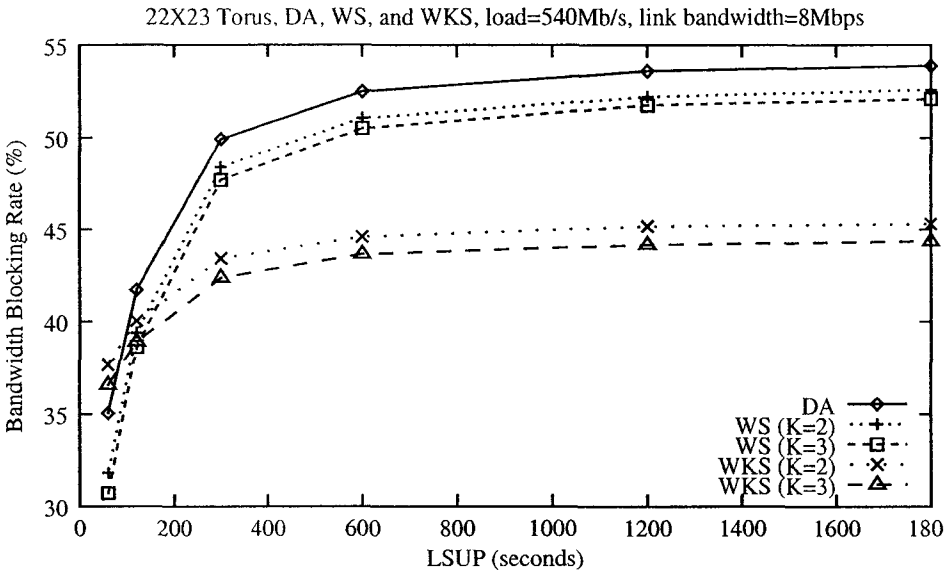


Figure 4.19: 22X23 torus, Blocking rate vs. LSUP of DA, WS and WKS

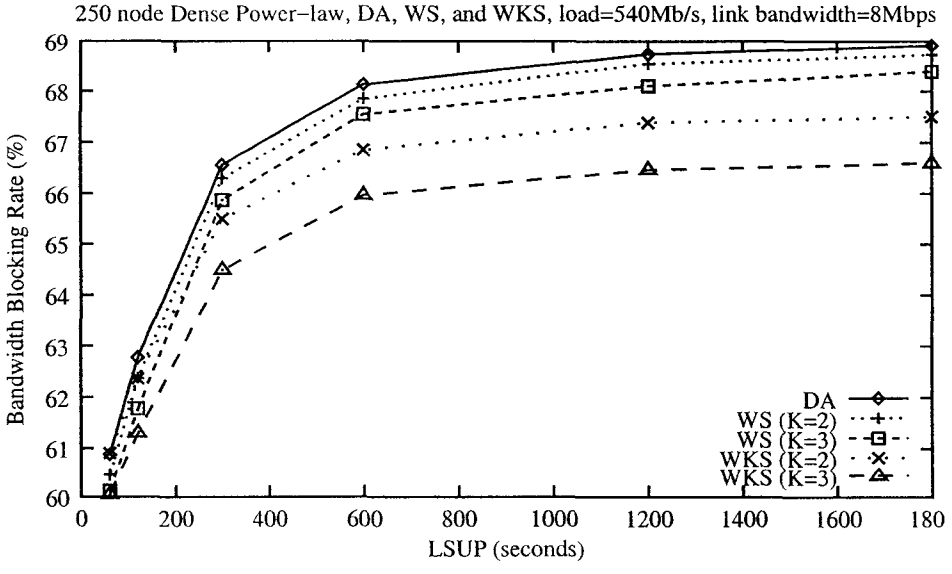


Figure 4.20: 250 node dense power-law, Blocking rate vs. LSUP of DA, WS and WKS

- the paths in the routing table are denoted as $path(i)$ ($1 \leq i \leq K$ for WKS and $1 \leq i \leq 2$ for DA),
- $bw(i)$ is the bottleneck bandwidth of the $i$th path,
- $hop(i)$ is the hop count of the $i$th path,
- $h_{min}$ is the hop count of the the minimum hop path, and
- $H$ is the total number of different hop counts for all the $K$ paths.

According to [49], a DA path is a "widest–shortest" path with no more than $n+1$ hops. DA actually is a multiple–path scheme with $K = 2$, $hop(2) - hop(1) = 1$, and $bw(2) > bw(1)$. In the case of $LSUP = 0$, if there exist paths with $h_{min}$ and $h_{min}+1$ hops, DA has two alternative paths to choose from. But the situation is more complex for WKS. When the paths are calculated based on accurate link state information, there are in fact only $H$, not $K$, paths that can possibly succeed. This is because for all the paths with the same length, if the widest route fails, the others will also fail due to their smaller bandwidth. Only if $H > 1$ can WKS have at least two alternative paths. Thus, if the link state information is accurate, WKS may occasionally provide fewer feasible paths than DA. However, this is not the case when $LSUP > 0$. For WKS, even if $H = 1$, all $K$ paths can possibly be successful. Besides, for DA, some paths are eliminated by the restriction $bw(2) > bw(1)$, and those eliminated paths could be feasible due to the inaccurate information. Thus, if $LSUP > 0$, WKS always provides $K \geq 2$ routes to select from while DA provides at most two. As illustrated in Figure 4.19 and Figure 4.20, WKS outperforms DA even when $K = 2$.

## 4.6 Conclusions

We have studied a collection of multiple–path routing schemes and investigated their performance. Our study considered a diverse set of traffic conditions and topologies (including realistic topology models) with the intention of uncovering the benefits and limitations of multiple–path routing schemes.

Our experiments have indicated that the proposed approach to routing offers a significant improvement over the single–path approach, especially if the link state information is outdated and inaccurate. Particularly, the proposed routing schemes trade insignificantly increased signaling overhead for significantly improved blocking performance and reduced link state exchange overhead, which is the major part of the total routing overhead. The experiments also show that the hop–based algorithms consistently win over the bandwidth–based ones. The best of our routing algorithms, Widest–$K$–Shortest, also outperforms the *dynamic alternative* and *widest–shortest* in inaccurate information environments.

We have demonstrated that multiplicity of paths to select from is one possible remedy to the problem of limited accuracy of the link state information, which is bound to haunt all networks of non–trivial size. Routing solutions that do not take this problem into account will poorly scale to large networks, in which the link state information cannot be updated and propagated too often. As it turns out, it is more important to have a choice at all than to be able to choose from a large selection. Consequently, in terms of computational complexity, our routing algorithms are comparable to those that prefer to stick to a single path. Overall, there is ample evidence that multiple–path schemes can be a scalable solution for QoS routing in environments with inaccurate link state information. Nevertheless, the schemes presented here for the path construction and selection are by no means the only way to consider K alternate paths in QoS routing. Moreover, despite the evidenced advantage of Widest–$K$–Shortest, other schemes may exist that perform equally well. At the same time, a variety of relevant issues become interesting to study. One example is whether it is possible to approach the performance of selection schemes that select between the shortest paths (as Widest–$K$–Shortest does), if, instead, we are forced to select between the widest paths due to administrative or contractual restrictions that prohibit a view of the topology (and hence of the hop count) of other network providers.

# Chapter 5

# Cost–Effective Routing Information Dissemination

## 5.1 Introduction

In Chapter 4, we proposed multiple path based QoS routing schemes to deal with the routing inaccuracy problem. In this chapter, we will approach the scalability problem from the point of view of link state dissemination mechanisms.

In traditional link state routing, routers use flooding to disseminate link state information. Figure 5.1 demonstrates the flooding procedure of a LSA originated from node $F$. In reality, for every LSUP, each router initiates the flooding of its LSAs associated with the link attached to it. As can be seen from the figure, flooding is a costly operation, with every piece of information reaching every corner of the network and the same link state information arriving at a node more than once from different paths. For example, node $B$ receives the LSA from node F for 3 times and node $C$ receives it for 4 times.

In this chapter, we will first tailor the flooding mechanism by adjusting the scope and frequency of link state updates and by qualitatively separating the update messages. Specifically, we will propose three schemes for reducing the update information, which are aimed at differentiating the priority of nodes of different distance and sending more updates to those nodes of higher priorities. Besides, we will exploit the qualitative impact of the link state updates on routing performance, and our results show that by differentiating the content

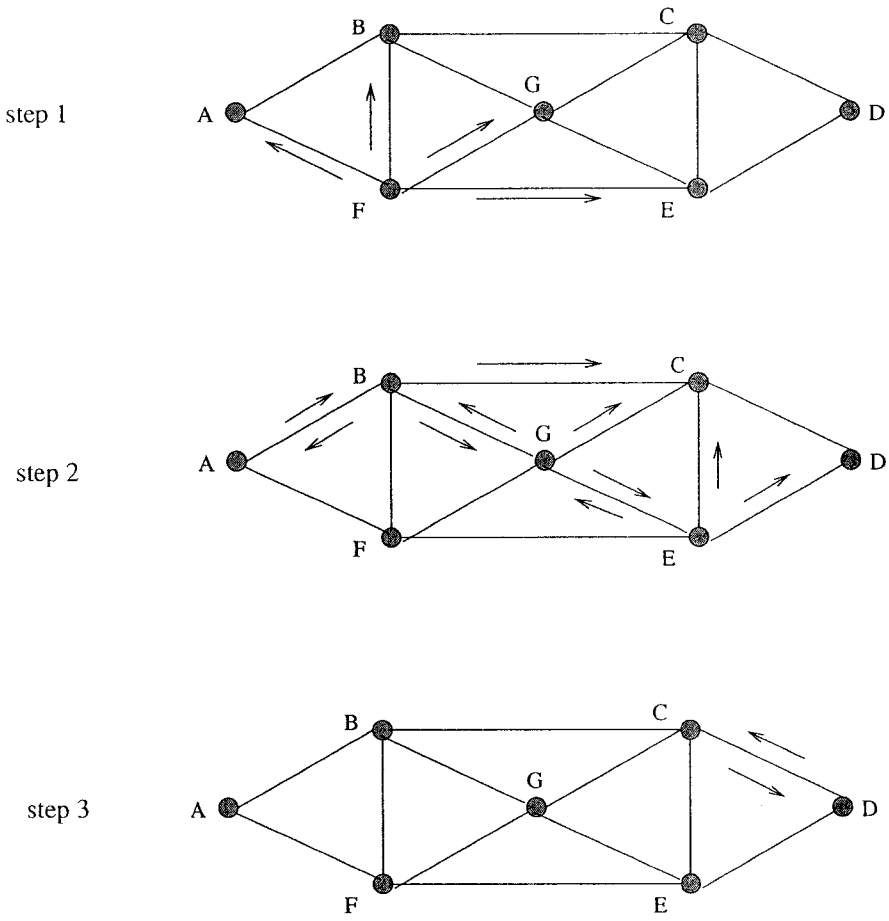


Figure 5.1: The flooding procedure of one LSA

of update messages, we can further reduce update costs without impairing routing performance.

## 5.2 Controlling Link State Dissemination

### 5.2.1 Distance–Based Link State Update Reduction

As introduced in Chapter 2, existing efforts towards reducing routing information distribution include update frequency reduction, which still depends on flooding, localized routing [56], which conducts routing without depending on globally exchanged information at all, and routing information aggregation/quantization [6, 26]. Confronted with these solutions, we raise two questions:

1. at a certain node, should the update information from all the other nodes be treated equally, as in the flooding scheme?

2. given that the local information about links adjacent to a node is accurate, should the information of the remaining links be completely ignored, as in [56]?

We think that the localized routing is extremely restrictive in that it only depends on the very limited local information; on the other hand, flooding is blind and redundant in that it generates a large amount of repetitive information. Therefore, in this section, we present an alternative that stands between the localized routing and flooding. Instead of ignoring non–local information at all, our schemes make use of it, but instead of treating updates from all the links the same, a node assigns different priorities to the received link state information based on how far their originating link is. In particular, the closer a link is to a node, the more important the value of the link's state becomes. The intuition supporting this approach has to do both with issues of spatial locality (the closer to a node a link is, the more likely it could be used by one or more paths that originate at that node) as well as less inaccuracy (the closer the link is to a node, the relatively faster the link state information will arrive).

We consider a distance–based update frequency reduction scheme, according to which link state information is sent to the closer neighbors more frequently. Specifically, we consider three variants of the scheme, the *Deterministic Frequency Reduction* (DFR), the *Probabilistic Frequency Reduction* (PFR), and the *Hop/Distance Threshold* (HDT).

The DFR and PFR are based on the same idea. Given an update source $s$, define the $h$–hop neighbors of $s$ as the nodes $h$ hops away from $s$. Supposing a node receives $m$ link state update messages from an upstream node, then it forwards only a portion $a$ $(0 \leq a \leq 1)$ of the $m$ messages to each of its direct neighbors, rather than forwarding all the $m$ message to them, as flooding would. That is, if there are $m$ update messages generated from $s$, then each of its $h$–hop neighbors only receives $m \cdot a^h$ messages, where $a$ is named as

the *frequency coefficient* (FC). The only difference between PFR and DFR is that in DFR, it is deterministically decided how many link state updates are forwarded to the direct neighbors, while in PFR, it is probabilistically determined. For example, if $FC = 0.8$, then with DFR, for every 5 link state updates received, a node forwards only 4 to its direct neighbors. With PFR, however, every time a node receives an link state update, it forwards the update to each of its direct neighbors with a probability of 0.8. Obviously, the extreme case of $FC$ being 1.0 is actually the case of flooding.

In the third scheme, the HDT, there are two update frequency levels. If the distance from a node $i$ to the update source $s$ is smaller than the hop/distance threshold, then node $i$ is updated once every LSUP time, otherwise, $i$ is updated once every two LSUP times. The idea is to update near nodes twice as frequently as far away nodes.

In terms of implementation, the proposed schemes require only minor modifications to OSPF. For example, for DFR with a $FC$ of 0.8, a router only need to keep a counter with an initial value of 5 for each neighbor. Every time the router receives a LSA from a neighbor, the router checks if the counter is larger than 1. If so, it would forward the LSA to its downstream nodes and decrease the corresponding counter by 1; otherwise, it does not forward the LSA to any of its downstream neighbors and reset the counter to 5.

### 5.2.2 Simulation Results

In the following, we will investigate the impact of the three schemes on network blocking rates and costs savings from flooding. The blocking rate is weighted by the connection bandwidth, as used in Chapter 4. The metric capturing the communication cost savings (in terms of link state message exchanges) relative to flooding can be defined as follows:

$$\text{Cost Savings } (\%) = \frac{C_{flood} - C_{DFR}}{C_{flood}},$$

where $C_{flood}$ is the costs in the case of flooding, and $C_{DFR}$ is the corresponding communication costs for DFR. Equivalently we define the cost savings of PFR and HDT.

We make use of the simulation model and the SKW scheme presented in Chapter 4, and observe the results of applying a scheme such as DFR on a regular torus topology. Since the results for a torus are not representative of real topologies, we simulated both a torus and a realistic power–law based random Internet topology [50]. The torus topology includes 16x16=256 nodes, while the power–law topology includes 250 nodes. All link bandwidths are set to 155 Mbps, unless otherwise indicated. Connection requests arrive at a rate controlled by the offered load parameter. The requested bandwidth for each connection is generated in a random uniform fashion between the values of 1 and 5 Mbps. The connection requests follow an exponential interarrival distribution. The connection holding time is log-normal with an average duration of 180 seconds.

First let us look into the results of applying a scheme such as DFR on a regular torus topology. Figure 5.2 and Figure 5.3 illustrate both the impact on the blocking rate as well as the impact on the costs (in terms of messages). The cost reduction for a FC less than 1.0 is dramatic because of the compounding effect from reducing the information as it propagates further away from its source. However, the blocking rate deteriorates only slightly for large values of FC. Indeed the maximum benefits for the least penalty in additional blocking are realizable for an FC of 0.9 in the case of DFR and torus topology. The results of PFR are very similar to those of DFR but the best value for FC is not the same as that for DFR.

Figure 5.4 provides a comparison of the blocking rate results for DFR, PFR and HDT. In all cases there appears to exist a value, either in the FC parameter, or in the hop threshold parameter (for HDT) that minimizes blocking when large values of LSUP are used. Specifically, in DFR and PFR, for large LSUPs, increasing the FC leads to improved blocking up to a point, beyond which any increase leads to a worse, but eventually converging, behavior. In HDT, the increase of the hop threshold first improves blocking performance, but after a certain point, it eventually results in a worse blocking behavior. These counter–intuitive observations are also true for non–regular, random power–law topologies. Figure 5.5 presents the results for a random 250 node

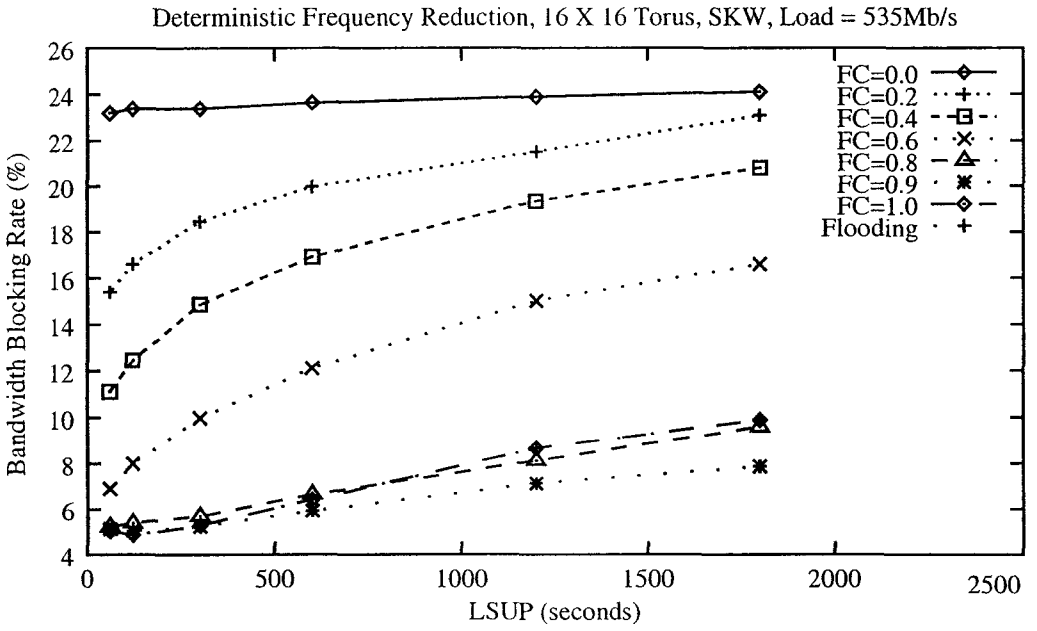


Figure 5.2: DFR blocking rate vs. LSUP in a 16x16 torus for different frequency coefficients (FC).

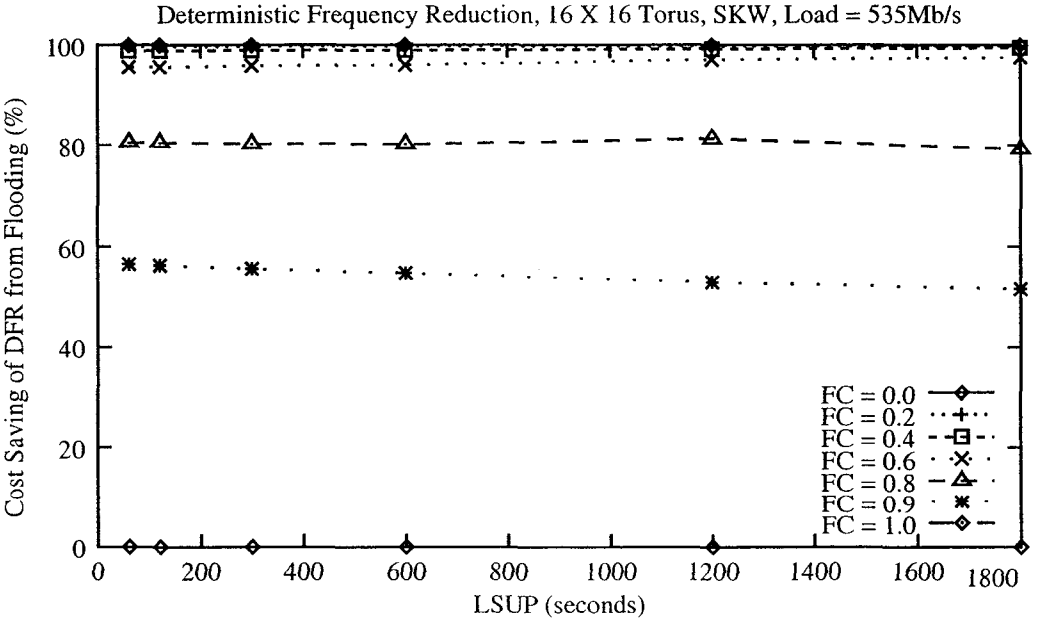


Figure 5.3: DFR cost savings vs. LSUP in a 16x16 torus for different frequency coefficients (FC).

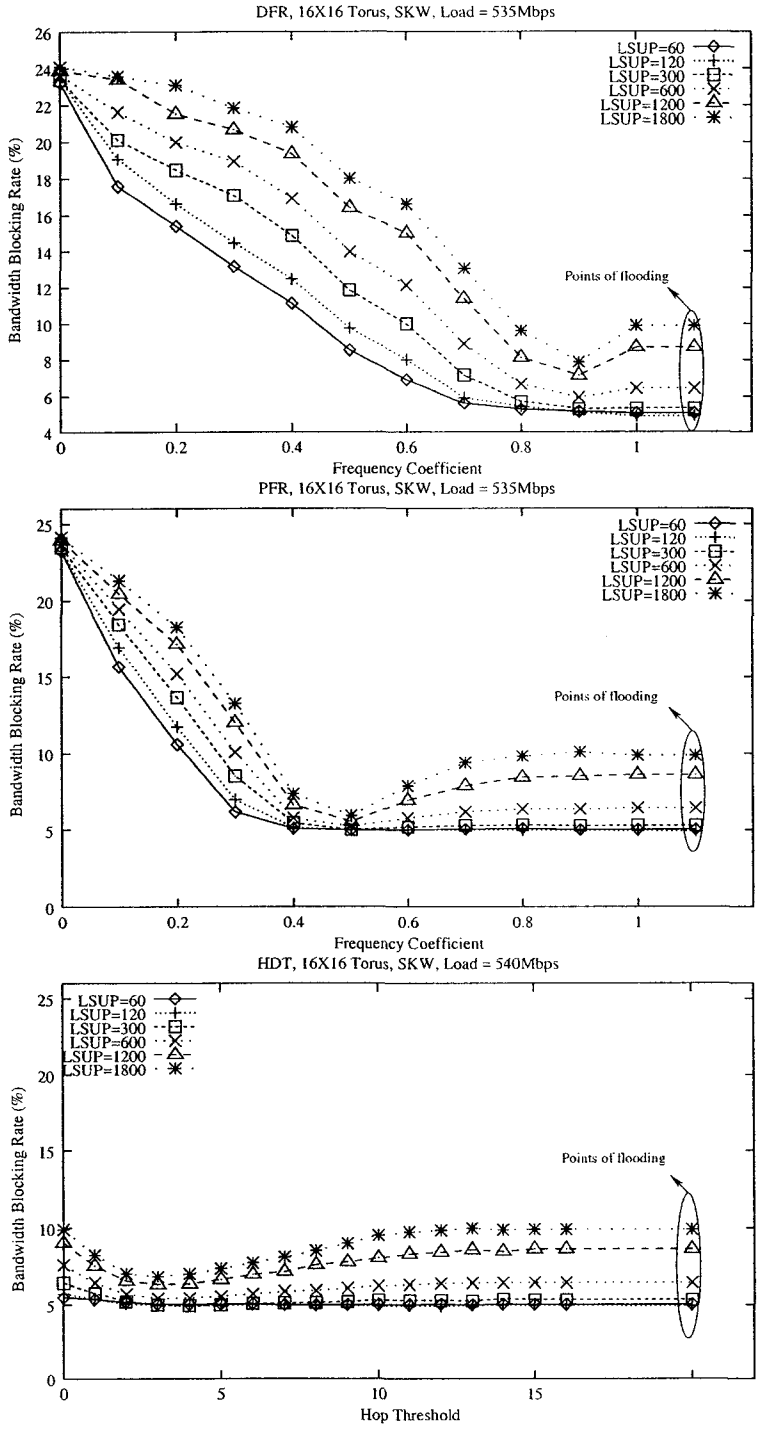


Figure 5.4: Blocking rate in a 16X16 Torus topology vs. frequency coefficient (for DFR) for different LSUPs.

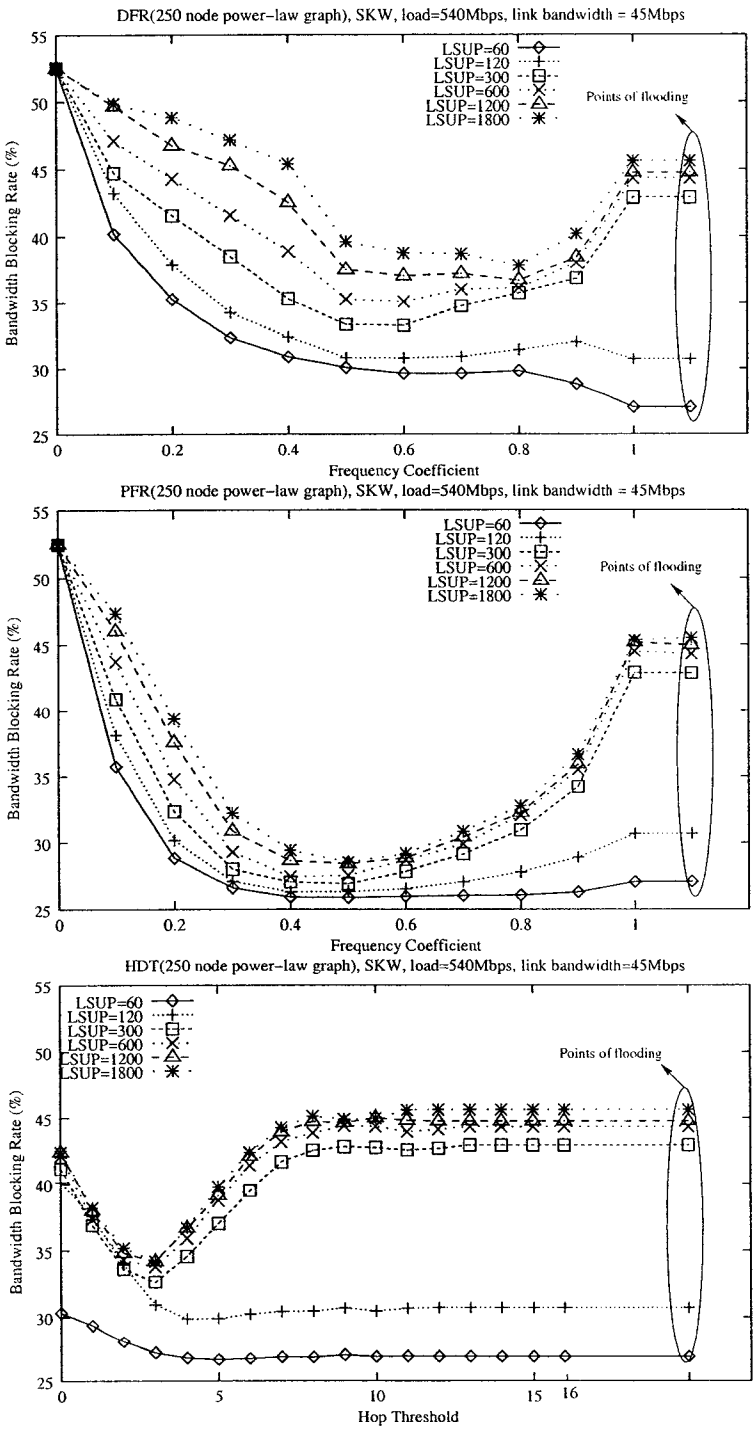


Figure 5.5: Blocking rate in a 250 node power–law topology vs. frequency coefficient (for DFR) for different LSUPs.

network following the topology construction in [50]. In fact, the results in realistic topologies suggest an even more dramatic behavior, which confirms the same observations we pointed out for the torus topology simulations.

Why would more information about the link state result in worse blocking in case of large LSUPs? We find that under the following scenarios we can observe high blocking rates:

1. small LSUPs (e.g., LSUP = 60 or 120 seconds,) with small FCs/hop–thresholds (i.e., few nodes get updated). The smaller the FC is, the higher the blocking rate is.

2. large LSUPs with small FCs/hop–thresholds. The smaller the FC is, the higher the blocking rate is.

3. large LSUPs with large FCs/hop–thresholds. The larger the FC is, the higher the blocking rate is.

The first two cases share a common feature that the number of nodes which receive updates is extremely small. As a result, during the long simulation period, only a small part of nodes receive fresh information, while a majority of them keep using the same piece of information to make routing decisions, which inevitably causes network traffic congestion at some hot–spot links and therefore causes high blocking rates. Small LSUPs can not help because, despite the fact that the updated nodes receive accurate information, the number of nodes which do not receive updates are still huge, causing biased load distribution.

Based on the aforementioned analysis, we can easily understand why case 3 results in a high blocking rate. Since the LSUP is very large, all the updated nodes maintain the old link state for quite a long time. In the case that a majority of nodes receive updates, i.e. large FCs/hop–thresholds, same facts as in the first two cases still hold. That is, a majority of nodes stick to a certain link state database for quite a long time, which brings about biased traffic distribution and congested links. Therefore, it is understandable why the performance is the worst when FC=1. Accordingly, we can also

understand why the performance turns better as the FC value is reduced (but not extremely small as in the first two cases). With smaller FCs, the fact is that, during the long time between two updates, quite some nodes make routing decisions based on the stale link state database (because they do not receive updates at the last update), which have two advantages:

**a)** compared with all the nodes holding the same piece of information (e.g., $FC = 1$) for a long time, some nodes having different information (e.g., $FC < 1$) results in the diversification of computed paths and therefore realize better traffic dispersion.

**b)** in case of large LSUPs, fresh link states (e.g., $FC = 1$) are no more accurate than stale ones (e.g., $FC < 1$, or some nodes do not receive new link state at the update time).

Observing case 3, one may ask why the performance becomes better as the FC value approaches 1 in the case of small LSUPs, as apposed to the case of large LSUPs. The reason is that, with small LSUPs, the dilemma with large FCs/hop–thresholds in both case **a)** and **b)** does not hold any more, therefore the performance is consistently better as the FC value turns larger. In particular,

**a)** in case of large FCs/thresholds, it is still true that a majority of nodes hold the same piece of information, but with small LSUPs, every node's link state database changes more frequently than in the case of large LSUPs, thus impeding the occurrence of congestion.

**b)** when the LSUP is small, fresh link states imply accurate information; consequently, the more nodes are updated, the better the performance is.

### 5.2.3 Summary

Our simulation results show that the proposed distance–based link state update reduction schemes provide a cost–effective solution for link state dissem-

ination. It is interesting to find that, with large LSUPs, we can obtain optimized routing performance without depending on a large FC or hop-threshold value. This observation shows the scalability feature of our proposed algorithm. Moreover, the proposed schemes only need minor modifications to the existing OSPF flooding mechanism; consequently, it is easy to implement.

## 5.3 Qualitative Impact of Link State

The results of Figure 5.4 and 5.5 suggest that when the LSUP is large enough, fresher information may cause worse performance. To observe the impact of large LSUPs to the performance, in the following, we will break the situation into four cases. For link $l$, denote the available bandwidth of this link at the time of last update by $B$. Some time after the last update, we receive a connection request specifying $b$ as the requested bandwidth. The actual bandwidth of link $l$ at that time, which we do not know exactly, is $B'$. Essentially, there are four possibilities:

1. $B < b$ and $B' < b$.
2. $B > b$ and $B' > b$.
3. $B < b$ and $B' > b$.
4. $B > b$ and $B' < b$.

Cases 1 and 2 are less interesting because they do not influence the correctness of the routing decision for the current request. Let us focus on cases 3 and 4 that quite drastically change our options regarding the link. In case 3, $l$ is considered infeasible, while in fact it is feasible, whereas in case 4 it is the other way around. The question is now this: what are the consequences of scenarios 3 and 4? We think that the negative impact of case 3 is more significant than that of case 4, because case 3, by underestimating the link capacity, disables a successful route, while case 4, by overestimating the link capacity, just enables an unsuccessful route. Specifically, if at some time $t$ you

find a link heavily loaded, i.e., infeasible for a typical request, then as connections on the link start terminating, the link tends to become less loaded. However, the released bandwidth can not be utilized until the next link state packet is sent, which is *not* in the near future if the LSUP value is fairly large. Therefore, we conjecture that it makes sense to propagate "good news" faster, by which we mean besides sending out updates periodically, to propagate an update whenever a heavily loaded link becomes less loaded. In doing so, we reduce the chances of case 3. In another word, we propose to qualitatively distinguish between update contents.

In this section, we will investigate the qualitative impact of link state information. We define a *qualitative* link state update dissemination scheme as a scheme according to which decisions to send new link state information is triggered upon a drastic quantitative change, sufficiently large to consider that the corresponding link has changed essentially in a qualitative sense. The qualitative changes monitored here are drastic increases and drastic decreases of the link bandwidth. In our study, any change of 30% of the link's bandwidth is considered "drastic". To examine the aforementioned conjecture, we review the performance of four information propagation schemes, dubbed *Good News* (GN), *Bad News* (BN), *Good and Bad News* (GBN) and *Periodic Link State Only* (PLS) as they apply to QoS routing using the multiple–path technique to compensate for inaccuracy. In GN, aside from periodic link state updates, there are also updates sent whenever the load *decrease* on a link exceeds the "drastic" threshold since the last value reported in link state messages. BN sends updates periodically as well as when the link load *increase* exceeds the threshold. By the same token, GBN conducts updates not only periodically but also when the *increase* or *decrease* of the load of a link exceeds the threshold. PLS is the regular periodic update scheme. In all cases, we keep track of the costs introduced by the exceptional messages necessary to convey the "bad" (or "good") news.

We investigate the blocking rates and link state exchange costs of GN, BN, GBN and PLS as functions of the LSUP and network load. Figure 5.6 reveals that when the LSUP is less than a certain point, the traditional PLS scheme

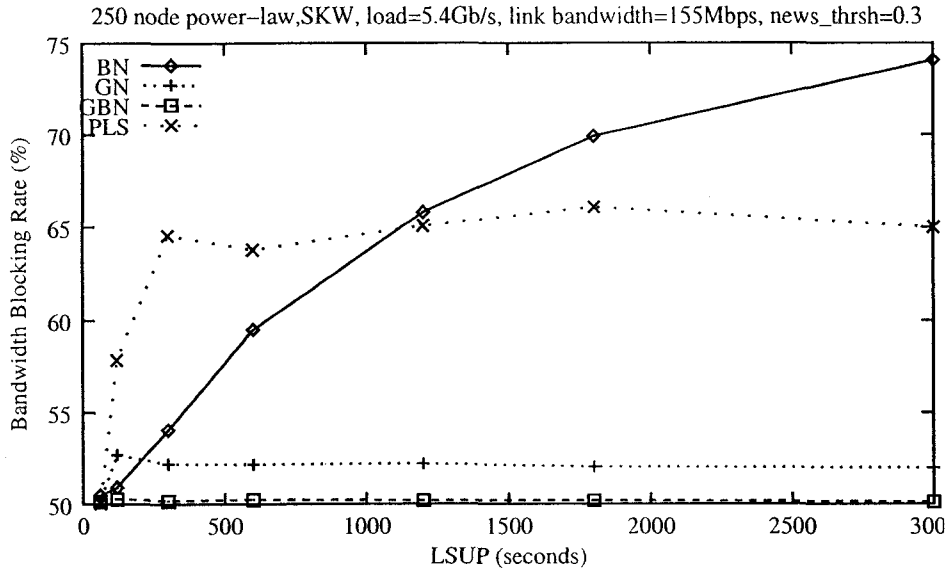


Figure 5.6: Blocking rate vs. LSUP on a power-law random topology of 250 nodes.

performs the worst compared with the other three. This is understandable because in case of a small LSUP, the other three schemes produce more up-to-date information in addition to periodic updates. Among the remaining three, GBN appears to perform the best, BN the worst, and GN is in-between, but the upside for GN is that it achieves a comparable blocking ratio to that of GBN with significantly less message overhead, as shown in Figure 5.6 and Figure 5.7.

The second observation is that the BN approach is definitely the worst for increasing LSUPs. The explanation is fairly straightforward. Between link state updates, there exist $K$ paths that can be used. If "bad news" arrives then chances are that we are going to restrict the number of feasible paths to something less than $K$. That is, we start with limited options that, along the way, are limited further, until the next link state update arrives. This is also the reason why BN turns out to be worse even than PLS.

We conclude by looking into the load-dependent performance of BN, GN, GBN and PLS. In Figure 5.8, Figure 5.9, Figure 5.10 and Figure 5.11, we vary the load and observe the blocking rate and the number of link state messages sent. It is clear that regardless of whether the LSUP is short or long, the two

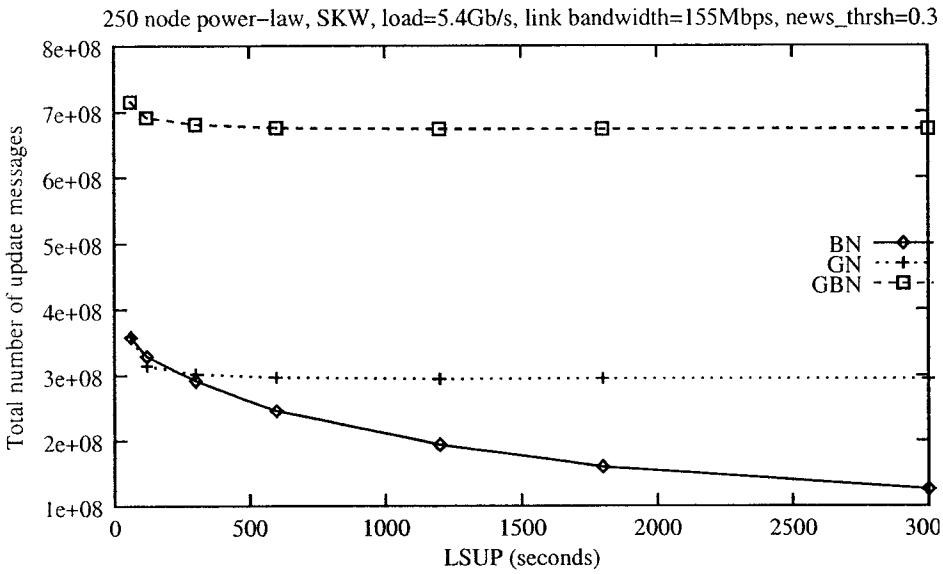


Figure 5.7: Cost vs. LSUP on a power-law random topology of 250 nodes.

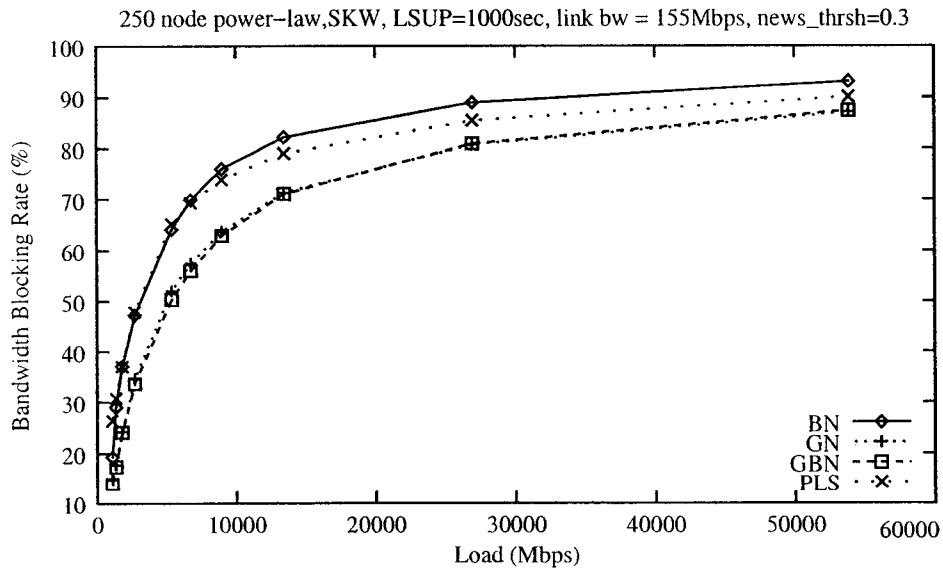


Figure 5.8: Blocking rate vs. network load for a LSUP of 1000 seconds. Power-law random topology of 250 nodes.

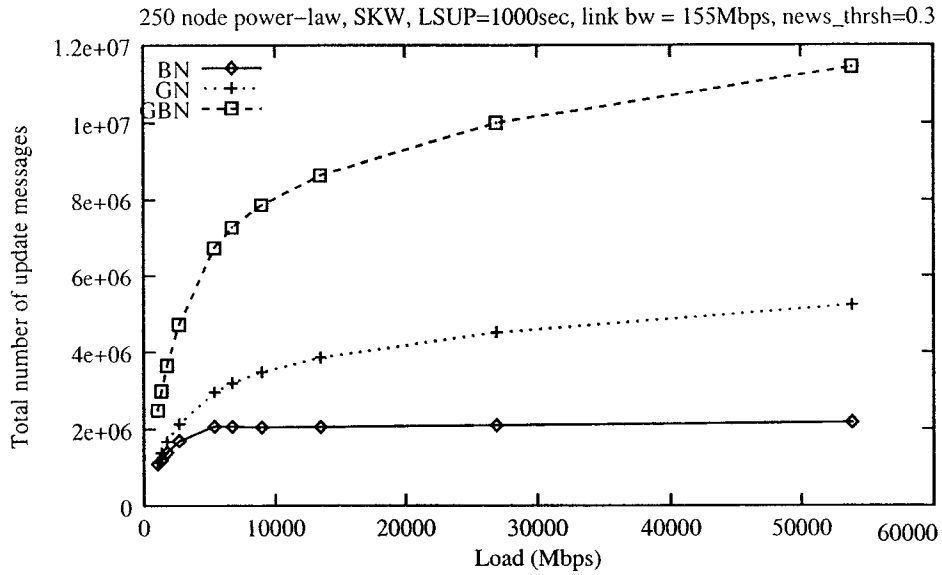


Figure 5.9: Message costs vs. network load for a LSUP of 1000 seconds. Power-law random topology of 250 nodes.

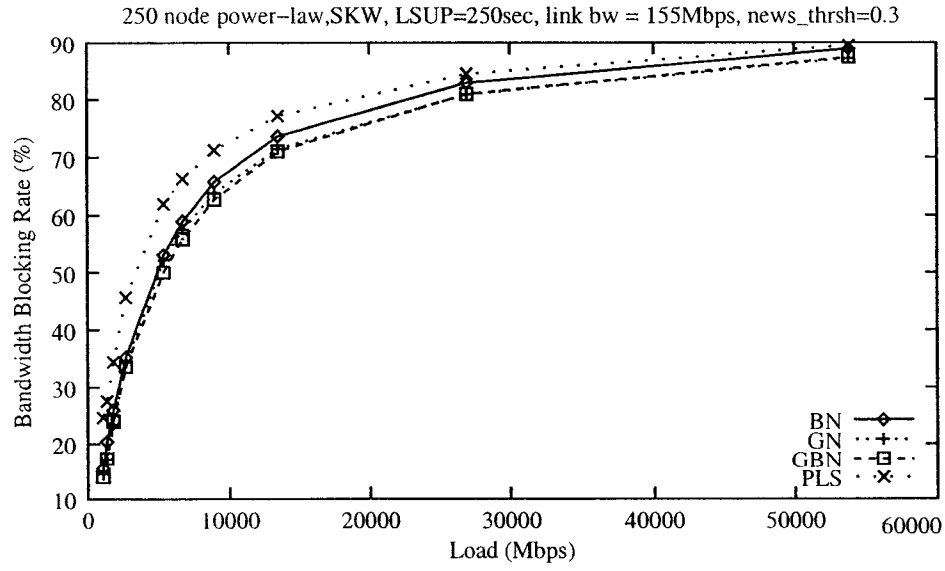


Figure 5.10: Blocking rate vs. network load for a LSUP of 250 seconds. Power-law random topology of 250 nodes.

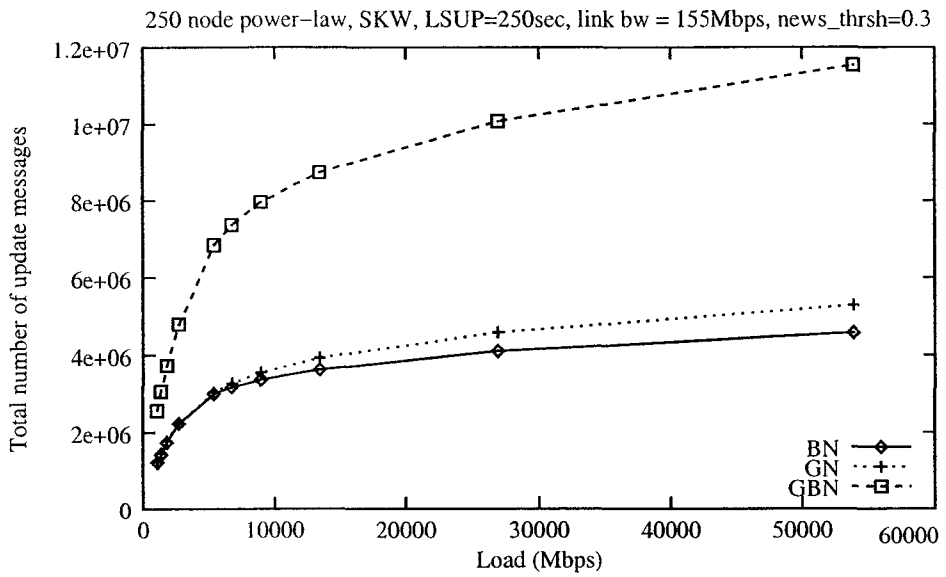


Figure 5.11: Message costs vs. network load for a LSUP of 250 seconds. Power–law random topology of 250 nodes.

schemes GN and GBN perform the best. BN's impact is not as detrimental, if the LSUP is sufficiently short, but for a comparable cost in terms of link state messages, GN guarantees a better behavior. For a significantly higher cost one can get the advantage of GBN. However, as Figure 5.8, Figure 5.9, Figure 5.10 and Figure 5.11 illustrate, the overall improvement cannot be considered spectacular. Nonetheless, smaller overhead for GN suggests that it is a reasonable compromise between the blocking rate improvement and the additional message overhead involved. The simulation confirms our intuition that "good news" should travel "faster". In other words, it is more important to learn when a loaded link becomes free than when a free link becomes loaded. After all, we are bound to discover the latter anyway if we attempt to setup a connection over it.

## 5.4 Conclusions

In this chapter, based on our previous work on $K$–path selection criteria, we attempt to establish cost–effective link state dissemination mechanisms. First we propose three distance–based link state update reduction schemes, DFR, PFR, and HDT, which are aimed at reducing the redundancy and "blindness"

of the traditional flooding mechanism by assigning different priorities to nodes based on their distance to the updating source. The proposed schemes appear to offer comparable performance with flooding while consuming significantly reduced update messages. Furthermore, we find a counter–intuitive observation that, with large LSUPs, the best performance points are not associated with the scenarios where all the nodes receive link state updates, but with the scenarios where some nodes keep their out–of–date link states. We argue that this phenomenon results from the traffic dispersion caused by partial nodes keeping stale information. This observation reveals the scalability feature of the proposed link state propagation mechanisms; also, it inspires our further thinking about the consequences related to large LSUPs.

As a result, we find that the frequency of link state dissemination is not the only factor to influence performance. For large LSUPs, the nature of the link state information also has a dramatic effect. We study the problem by considering exceptional additions to the periodic link state updates, when the residual link bandwidth decreases or increases dramatically. We subsequently examine the resulting blocking rate improvement and its corresponding control message overhead. Our findings indicate that the "good news" strategy, i.e., the one to propagate extra bandwidth increase information, is the most cost–effective routing performance boosting techniques among all the alternatives.

# Chapter 6

# Deflection Routing

## 6.1 Introduction

In previous chapters, we have approached the QoS routing problem using multiple paths based on the assumption of a reservation–based model. According to this model, all packets of an end–to–end session follow exactly the same path from source to destination. Intuitively, the deterministic character of such a connection makes it easier to set aside the right amount of resources at every intermediate node and predict what is going to happen when several virtual circuits pass at the same router. Consequently, most work on QoS–driven resource allocation focuses on path selection algorithms [1, 38, 47, 73], assuming that once selected the (single) path will be followed by all packets of the session. This approach essentially equates a transport–layer session with a network–layer virtual circuit, even if (as in the Internet) the network–layer virtual circuit is not explicit.

While moderate attempts at multi–path routing schemes found in the literature [6, 21, 63, 38] do demonstrate that a better load balancing can be achieved this way, they still restrict the selection of alternative routes based on a more or less explicit notion of a network layer session. This often tacit assumption about the inherent superiority of virtual circuits over datagrams has resulted in a complete oblivion of forwarding ideas based on deflection [13, 53] (which can be viewed as an extreme implementation of unkempt spontaneous routing) and has brought us the paradigm of ATM networks, which, until not so long ago, was viewed by many authors as the ultimate solution to all

problems of networking.

Even if one was to consider a pure datagram network–layer protocol, such as IP, where virtual circuits are absent, one can still identify a strong insistence on deterministic–path forwarding. As keeping track of TCP sessions in the network (IP) layer is neither easy nor natural, this insistence practically removes all flexibility from the transport layer. This is because, with the splitting of a single session over multiple paths considered harmful, there seems to be no alternative to forwarding all IP traffic between a given host pair via the same route. This in turn effectively kills all opportunities for load balancing.

Moreover, a conceptually similar approach to ATM virtual circuits survives to this day in the guise of MPLS [23] whereby bundles of transport layer flows are grouped (for the sake of efficiency) together in logical virtual circuits and are routed together. Unsurprisingly, MPLS further limits the potential for load balancing that could exist between transport layer flows. While the potential for load balancing at the level of MPLS "bundles" still exists, the fact that it can only be accomplished by (costly) routing decisions and routing algorithm (re–)calculations, results in MPLS being less than attractive for responsive load balancing.

In this chapter, we consider a flexible routing model based on deflection, whose degree is affected by the amount of buffer space available at the router. Our objective is to investigate how the QoS perceived by a transport–layer session depends on the way the buffer space available globally in the network is partitioned among the routers and the destinations. We conclude that with a global view of network resources, multiple alternative paths explored by different packets of the same transport–layer session need not be harmful. Just the opposite, they may in fact improve the utilization of those resources and, *at the same time,* improve the critical QoS characteristics of isochronous sessions. This is in contrast to what most people seem to believe. While one can easily agree that the increased routing flexibility naturally translates into a more balanced utilization of network resources, realizing that this approach may also imply better (more predictable) end–to–end delivery is a less obvious (and somewhat counterintuitive) step to take. This seems to confirm the

speculations expressed in [18] and suggests that single–path forwarding is an inherently flawed routing strategy.

The rest of the chapter is organized as follows: Section 6.2 introduces the basic design tradeoffs related to the placement choice for storage (buffer space) in the interior of the network versus the periphery. Section 6.3 provides a simplified network model for the sake of exploring the tradeoffs using a quantitative framework. Section 6.5 reviews the relevant simulation results and observations about the tradeoff and the comparison between deflection routing and conventional shortest path routing. Finally, conclusions and avenues for further research are summarized in Section 6.6.

## 6.2 The Tradeoff

Consider a router within the network core that is about to forward a packet belonging to a transport–layer session. Regardless of the assumed routing paradigm, the complete list of options regarding the fate of this packet consists of the following possible outcomes:

1. The packet is queued for transmission on the "best" output port (offering the "most attractive" route to the destination).

2. The packet is dropped, e.g., because of the lack of buffer space at the router.

3. The packet is queued for transmission on an output port that is considered a secondary choice (by the assumed route preference scheme).

With the single–path forwarding paradigm, the third possibility is excluded. The optimization effort regarding the utilization of network resources is thus directed towards a precise description of what is meant by the "best" route to the destination, as well as determining the right packet scheduling policy at the router. The latter can be interpreted as part of the buffer management scheme, as it also prescribes the packet dropping rules.

If the third option is admissible, an alternative to dropping a packet (or sending it over the congested preferred path) is to forward it via a suboptimal

route. The most serious consequence of this decision is that the packet may (legitimately) arrive at the destination out of order. Thus, to consistently play back the received packets as fragments of the transport–layer session, the recipient must use a reassembly buffer [20, 58, 59] to re–order and possibly delay packets arriving out of schedule.

Consider a session with some specific QoS expectations carried out between a source $S$ and destination $D$. Suppose that the global forwarding scheme of the network excludes option 3, i.e., the session path is fixed. A router $R$ along the session's path may drop a packet if it runs out of buffer space, but all packets eventually delivered to $D$ are going to arrive in order. Consequently, $D$ needs no reassembly buffer to play the session back, although, depending on the session type, it may still need some buffer space to smooth out the jitter caused by variable buffering delays at the routers. Larger buffers at the routers will result in a lower packet dropping rate perceived by $D$, although they may increase the jitter, which, at least for some session types, may render the packets useless upon arrival. Depending on the scheduling policy (or policies) adopted by the routers, late packets may also be identified (and dropped) before they reach their destinations.

If option 3 is admissible, packets may arrive at $D$ out of order, and $D$ may have to reassemble them, for which task it may need some extra buffer space. But with this option, $R$ is able to carry out its duties with less buffer space than in the previous scenario. This is because now $R$ has an alternative to dropping a packet. Consequently, it is possible that the reduced amount of buffer space at the router will be compensated by the increased amount of buffer space at the destination.

In [18], it is argued that deflecting instead of dropping may be a fundamentally better approach from the global viewpoint of network resource management. First, managing the private per–session playback buffer at $D$ is considerably simpler (and better defined as a problem) than managing the shared buffer space at $R$ in the face of multiple and diverse sessions passing through the router and (possibly) its multiple scheduling policies. In contrast to $R$, $D$ applies the buffer to a single session at the exact point where its

delivered QoS parameters can be monitored with ultimate fidelity and authority. Thus, it can easily, consistently, and meaningfully adjust the buffer size to compensate for occasional fluctuations of the perceived QoS measures. Second, if the session can put up with packets arriving out of order (e.g., the packets can be processed as independent datagrams), $D$ does not have to bother with reassembly buffers, while $R$ would still try to "fix what ain't broke" and buffer the packets in its effort to provide for (unneeded) in–order delivery. This is because $R$ doesn't know any better: even if it differentiates some elements of the service (the scheduling policy), this one element (i.e., the individual route of a datagram) offers no degree of freedom.

## 6.3 The Routing Model

Our experiments reported in this dissertation can be viewed as the first step aimed at putting the above speculations on a formal ground. As the routing approach in our network model, we use asynchronous deflection, somewhat similar to that described and analyzed in [28], but admitting limited buffers at the routers. In our model, no packet is ever dropped at a router. When a packet arrives for forwarding and there is no buffer space available to queue it for transmission on the preferred output port, the packet is directed to an alternative output port instead of being dropped. This way, some packets that would have to be dropped by a conventional router are now likely to reach their destinations via alternative paths.

Each node ranks its repertoire of alternative paths using the approach described in [28], and limiting the choice to 4 paths. In essence, it calculates the four shortest first–hop–disjoint paths to each destination, with the shortest path considered most attractive. Obviously, to offer alternatives from the viewpoint of routing, the different paths cannot share their first hop.

The buffer space available at a router is partitioned among all the output ports, such that each port is assigned the same fixed amount. Every time a packet arrives at the router, it will be directed to the best output port with available buffer space. Although no packets will ever be dropped in this model,

the late arrivals of excessively delayed packets will render them useless at the destinations.

One may be somewhat concerned about the inflexibility incurred by the rigid partitioning of buffer space at the router. This approach simplifies the model and seems to be acceptable in a scenario involving regular network configurations. Besides, one can easily see that the conclusions from our experiments cannot be reversed (and are likely to be amplified) when a more flexible buffer allocation scheme is used.

Each node in our network behaves both as a router and a host, i.e., a source and/or a destination of a traffic session. The total amount of buffer space in the network is equal to $B + 4b$, where $B$ denotes the amount of space assigned to the destinations (to be used for reassembly buffers) and $b$ stands for the amount of storage available at each output port at the routers. Intentionally, $B + 4b$ remains fixed in a given experimental setup, while the ratio $B/b$ determines the adjustable balance between the two categories of storage.

## 6.4 The Simulation Model

We designed and implemented a packet–level simulator. Each node has four input and outgoing queues. Application packets are generated at a source node and inserted into an application queue. In an intermediate router, a background queue is maintained for the newly generated background traffic. For each forwarding interval, one packet, which could be an application packet or a background one, is fetched from each of the input queues to be routed and sent to the corresponding output queue; only when there is no packet waiting in a certain input queue can an external background packet be processed. Doing so avoids the overflow of one or more of input queues and guarantees the lossless feature of the network.

We consider perfectly regular 4–connected $5 \times 5$ and $8 \times 8$ torus networks respectively. All links have the same bandwidth of 1Mbps. For the length of links, we consider two cases: the even link length case, where each link has the

same length, and the biased link length case, where link lengths fall into four categories. Specifically, we use $a*b^n$, $(0 \leq n \leq 3)$, where $a$ and $b$ are simulation parameters, to represent the link length of a category. For the even link length case, $b$ is set to be 1, (e.g., $a = 1000$km and $b = 1$,) while for the biased link length case, $b > 1$, (e.g., $a = 100$km and $b = 5$.) The biased link length case creates larger delay variance than the even link length case, and thus we are able to observe the buffer tradeoff in different delay variance situations without resorting to a random topology. Similar results can be obtained from both of these cases. While such networks may seem large, one should notice that geographically smaller networks can only be more advantageous for deflection (owing to a smaller variance in multi–hop propagation delays), which will result in more pronounced conclusions regarding the observed tradeoffs.

The network caters to an isochronous application described by a Pulse Code Modulation (PCM) voice traffic model, whereby 53–byte packets (corresponding to ATM cells) are sent at the average rate of 64Kbps. Their actual arrival process is Poisson. The reason why we use the Poisson process to model the voice packet arrival is because this model requires minimum assumption. We look at the behavior of a selected source/destination (S/D) pair involved in an isochronous session, with the remaining nodes contributing uniform as well as biased background traffic with exponentially–distributed session arrivals. For the uniform background traffic situation, all the non–S/D nodes are uniformly selected to serve as session source and destination. For the biased background traffic situation, two sub–sets of nodes are selected, serving as a source and destination set respectively. The source (or destination) of a background session is uniformly generated from the source (or destination) set. Actually the uniform background traffic can be seen as a special case of the biased background traffic, with both the source and the destination set being all the non–S/D nodes. The simulated time period is $400,000$ms, corresponding to over $60,000$ packets. Six independent experiments are run for each data point.

Our simulator keeps track of several performance measures related to the voice session.

- The *loss rate* is equal to the ratio of the number of packets discarded at the destination to the total number of packets transmitted by the source. In the context of deflection routing, since a packet cannot be dropped at a router, loss can only occur at the destination—in two possible scenarios. First, it may happen that when the packet arrives, the reassembly buffer is full and there is no way to store the packet until its scheduled playback time. Second, the packet may arrive too late, i.e., after its playback time, in which case the reassembly buffer cannot help, even if storage is available.

- The *network delay* of a packet is measured as the interval separating the packet's generation at the source and its arrival at the destination.

- The network delay is composed of the *queuing delay* and the *propagation delay*, the latter of which also includes the *(re)transmission delay* experienced by the packet.

- The *playback lag* represents the time elapsing after a packet arrives at the destination and before it is played back. It reflects the pure impact of the reassembly buffer.

- The *end–to–end delay* captures the overall processing time of a packet within the network counting from the moment the packet is generated at the source, until its playback at the destination. It is the sum of the network delay and the playback lag.

- The *jitter* is defined as the standard deviation of the network delay.

- The *average deflection number* is defined as the average times of deflection for each packet. This parameter reflects the degree of deflection.

## 6.5 Results

In this section, we will investigate the router and destination buffer allocation tradeoff. Besides, we will compare the DeFlection (DF) routing strategy, as

described earlier in Section 6.3, with the conventional Single Path (SP) routing strategy, wherein packets are routed only along the shortest path and dropped in case of buffer overflow.

### 6.5.1 The Buffer Allocation Tradeoff

The highly consistence of the results allows us to present only a small fragment of the large collection of results from our simulation experiments. In the following, we will primarily illustrate our observations with the $5 \times 5$ network of even link lengths.

Figures 6.1, 6.3, 6.5, 6.7, 6.9, 6.11, 6.13 and 6.15 are the results obtained from biased background traffic, and Figure 6.2, 6.4, 6.6, 6.8, 6.10, 6.12, 6.14 and 6.16 are obtained from uniform background traffic. Although the biased traffic situation is more realistic than the uniform one, we investigate the routing behavior in both traffic situations to obtain a complete view of the routing performance.

The following figures show how the observed performance measures in the network depend on the partitioning of the global buffer space between the routers and destinations. Every single curve corresponds to a specific fixed total amount of buffer storage $(B+4b)$ and is a function of its allocation $(B/b)$. Also, comparisons are provided between the DF and SP routing strategies.

Figure 6.1 depicts the loss rates of SP and DF under biased background traffic in $8 \times 8$ and $5 \times 5$ torus respectively. Figure 6.2 is the uniform traffic counterpart. We can see that the observed drop rate tends to decrease as the mass center of the buffer space is shifted from the routers to the destination, then flattens for a while, and finally increases sharply. [1] What we see is two counteracting phenomena in action. The reduced amount of buffer storage at the routers results in more packets being deflected (and misordered) (see Figure 6.3 and 6.4), while the increased size of the reassembly buffers compensates for the misordering and makes it possible to reconstruct the session

[1] The sharply rising tails are not plotted in the biased traffic graph, because the the number of packet arrivals at the destination is too few (implying a large loss rate) to surpass the playback threshold and therefore the loss rate is not measured by the simulator.

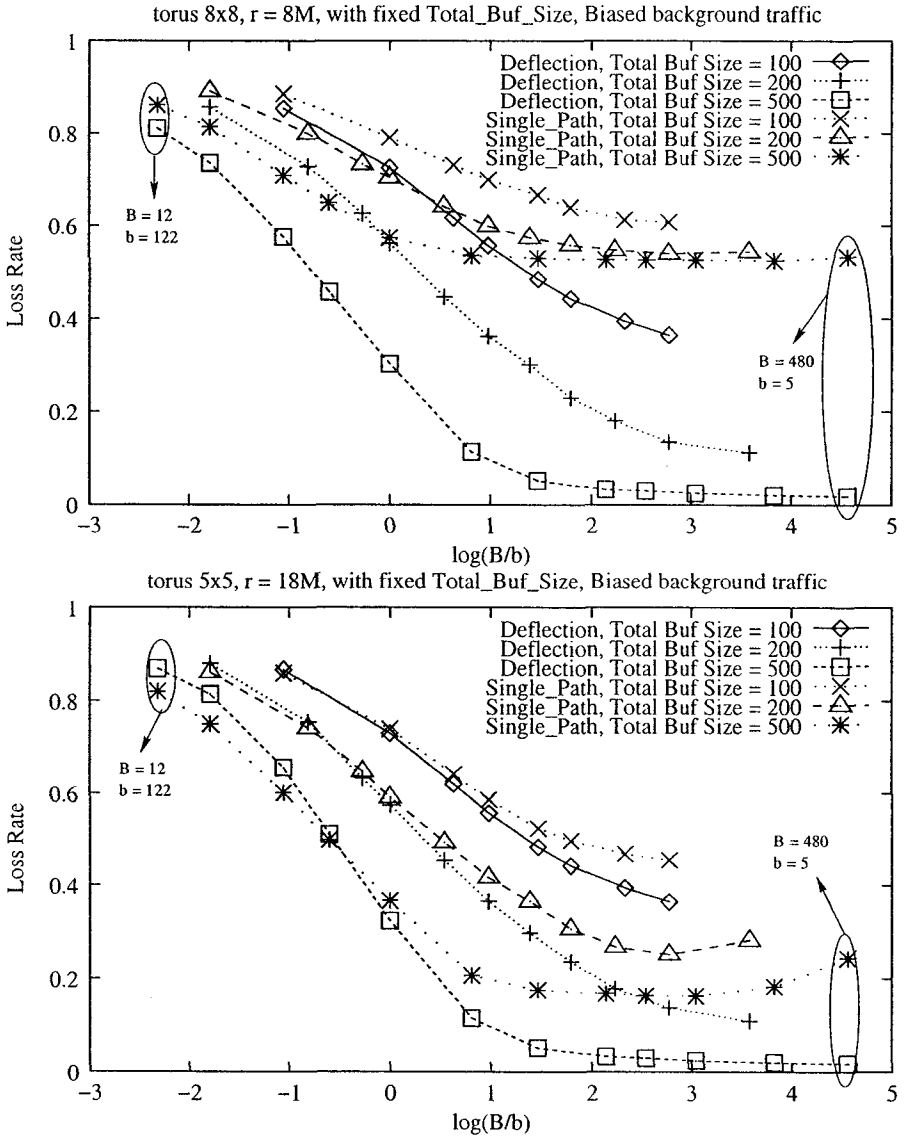


Figure 6.1: Loss rate vs. log(B/b), Biased Load.

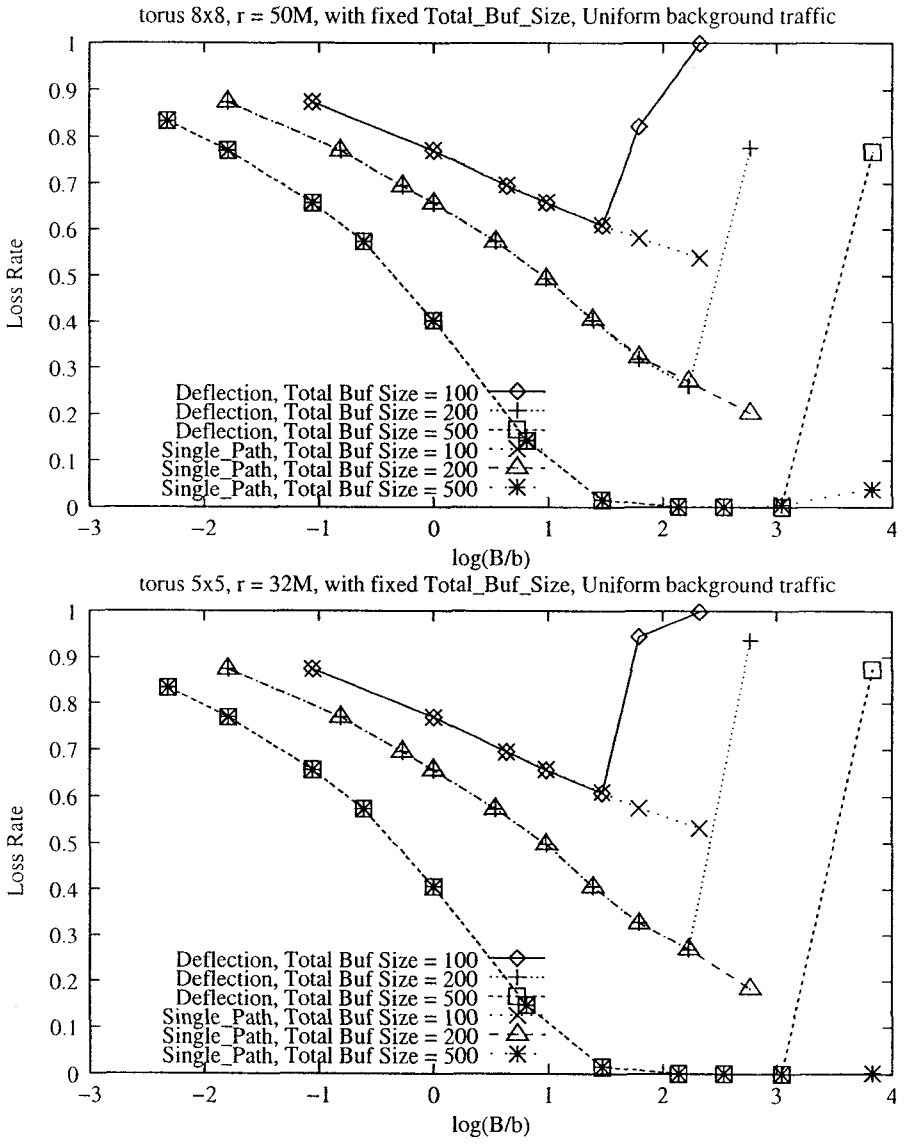


Figure 6.2: Loss rate vs. log(B/b), Uniform Load.

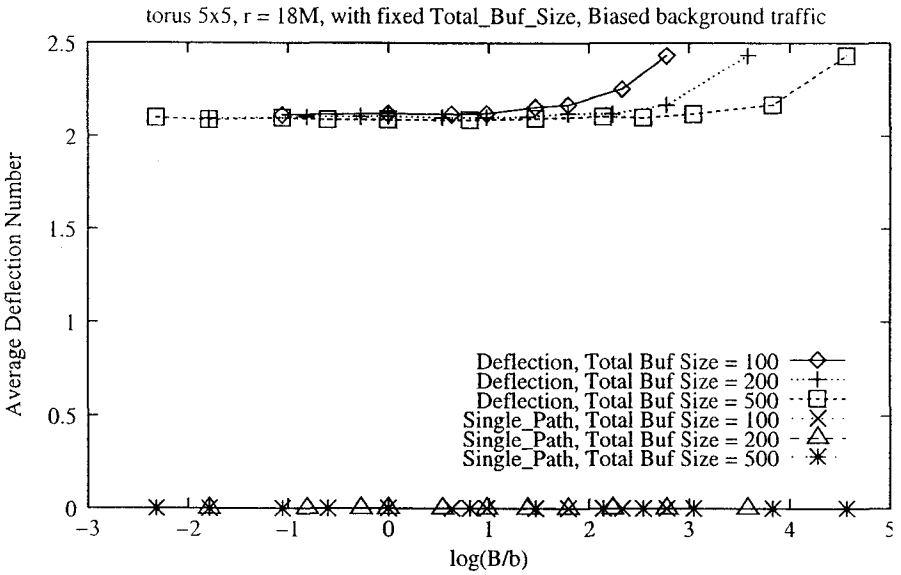


Figure 6.3: Average Deflection Number vs. log(B/b), Biased Load.

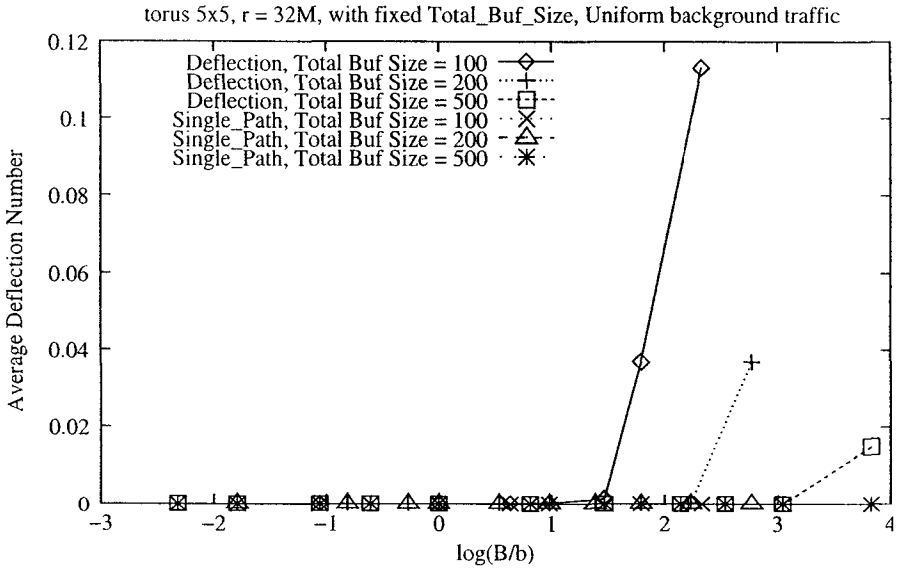


Figure 6.4: Average Deflection Number vs. log(B/b), Uniform Load.

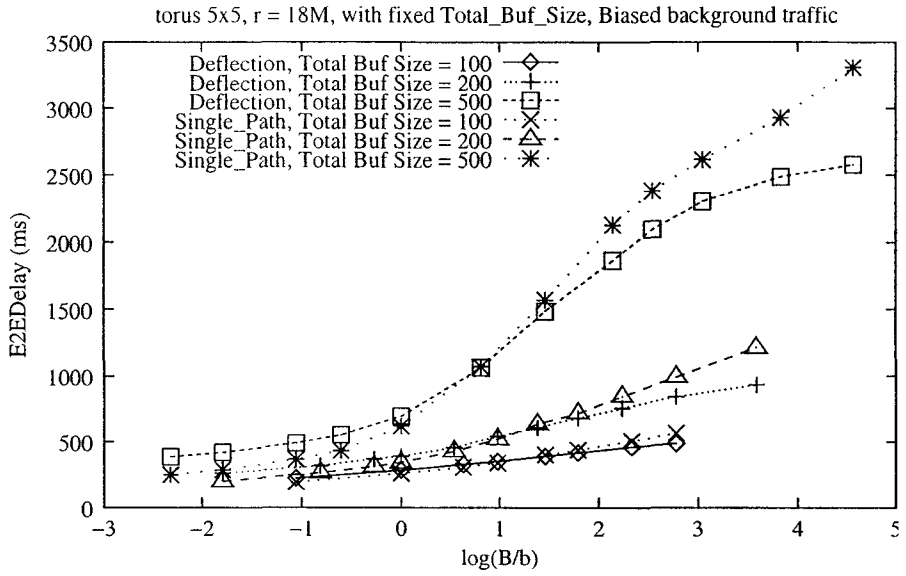


Figure 6.5: End–to–end Delay vs. log(B/b), Biased Load.

without dropping packets. It appears that below a certain $B/b$ threshold [2] this compensation is more beneficial than the increased level of deflection is harmful. For example, according to Figure 6.1, with a reasonable total buffer size, e.g., 500 packets, a workable low–loss regime falls roughly within the range $0 < \log(B/b) < 5$, which translates into $1 < B/b < 148$. This means that we have a large selection of $B/b$ resulting in an acceptably low loss rate, which widens as the total buffer space $(B + 4b)$ becomes larger. For the SP scheme, we can also observe the tradeoff, but to a less pronounced degree. The DF outperforms the SP in terms of loss rates, and the performance gap increases as the $B/b$ value goes up. This implies that deflection routing better suits the cost–effective buffer allocation scheme — large assembly buffers plus small intermediate router buffers.

The negative impact of the reassembly buffer on the QoS measures perceived by the voice session consists in increasing the end–to–end delay, as shown in Figure 6.5 and 6.6. The end–to–end delay is composed of the network delay (Figure 6.7 and 6.8 ) and the playback lag (Figure 6.9 and 6.10). According to these four graphs, the playback lag is the dominating factor, and therefore we observe an increasing end–to–end delay. The playback lag

[2] Same as 1.

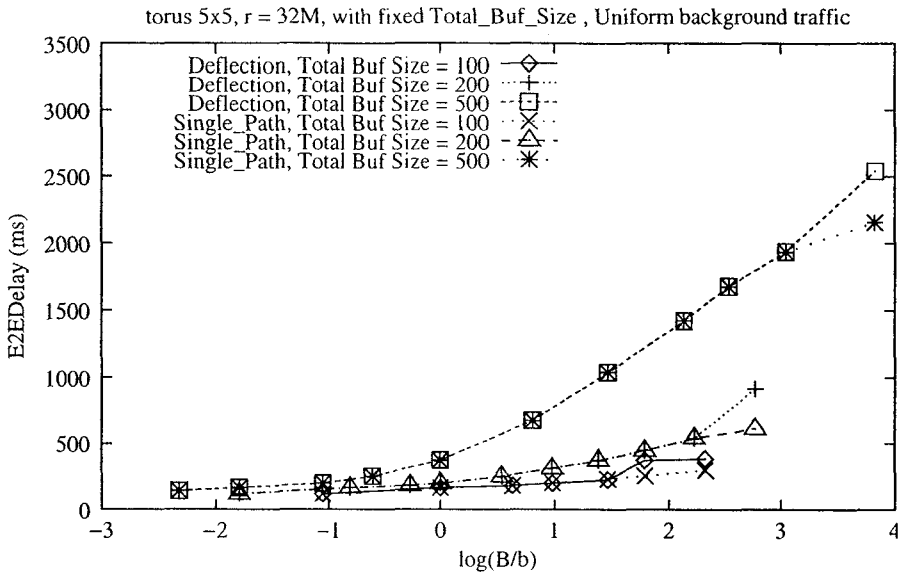


Figure 6.6: End–to–end Delay vs. log(B/b), Uniform Load.

increases along with $B$ because the destination buffer has to be filled up to a certain fraction of its total capacity before playback can commence. (This fraction was set at 80% based on the previous work [58].) The decreasing trend of the network delay is not surprising because, between its two components, the queuing delay (Figure 6.11 and 6.12 ) and the propagation delay (Figure 6.13 and 6.14), the former is the dominating factor. Obviously, the falling queuing delay curves are attributed to the reduced intermediate buffer size.

Another noticeable observation about the end–to–end delay metric is that the DF scheme exhibits a lower end–to–end delay than the SP scheme. Tracking the simulation unveils the reason: the playback points of the voice packets in the DF are earlier than in the SP, or the DF surpasses the playback threshold earlier than the SP. This is because deflection allows packets to be diverted to less congested network areas, increasing packets' chances to reach destination and avoiding the destiny of being dropped as in the SP case. Also, it implies that under a certain load condition, the DF scheme soothes network congestion more effectively than the SP scheme.

In the following, we will only focus on the loss rate and the end–to–end delay because these two QoS metrics are directly perceived by the end users.

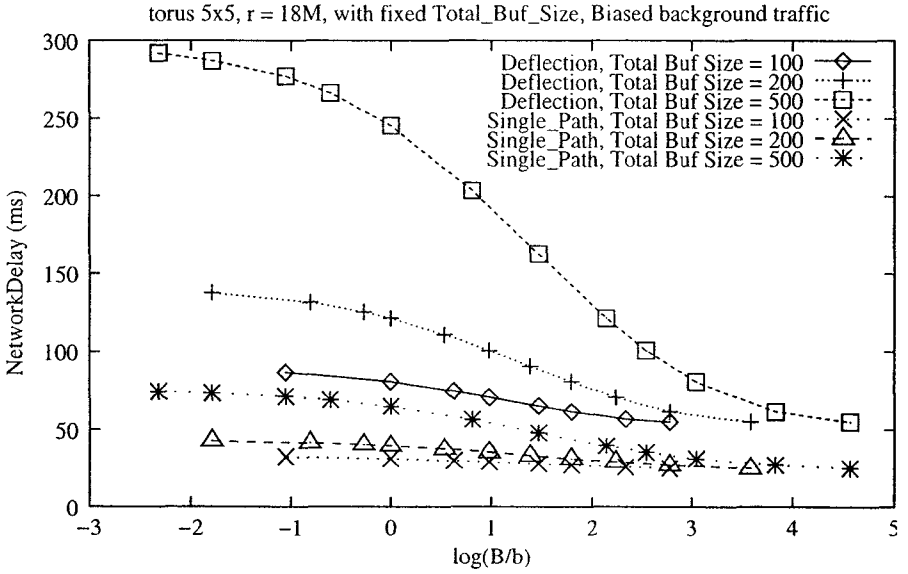


Figure 6.7: Network Delay vs. log(B/b), Biased Load.

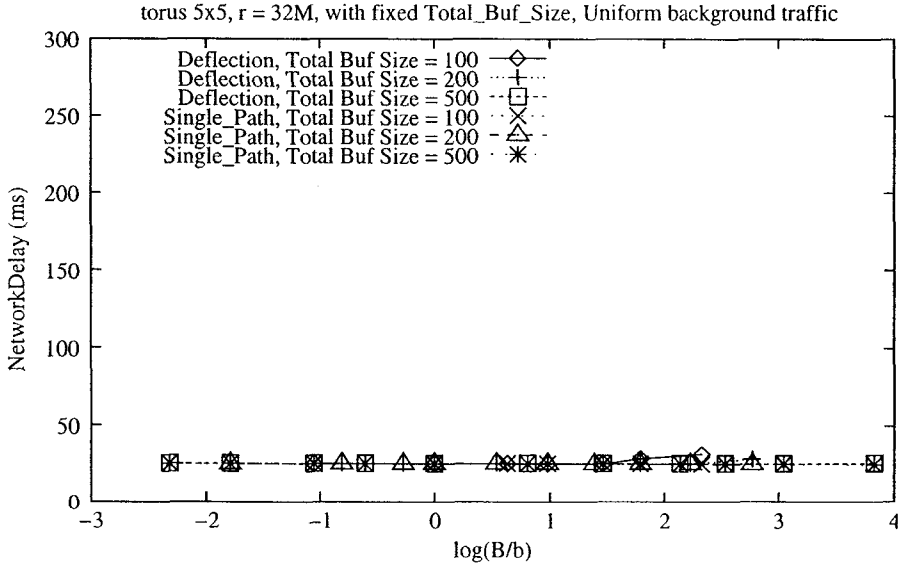


Figure 6.8: Network Delay vs. log(B/b), Uniform Load.

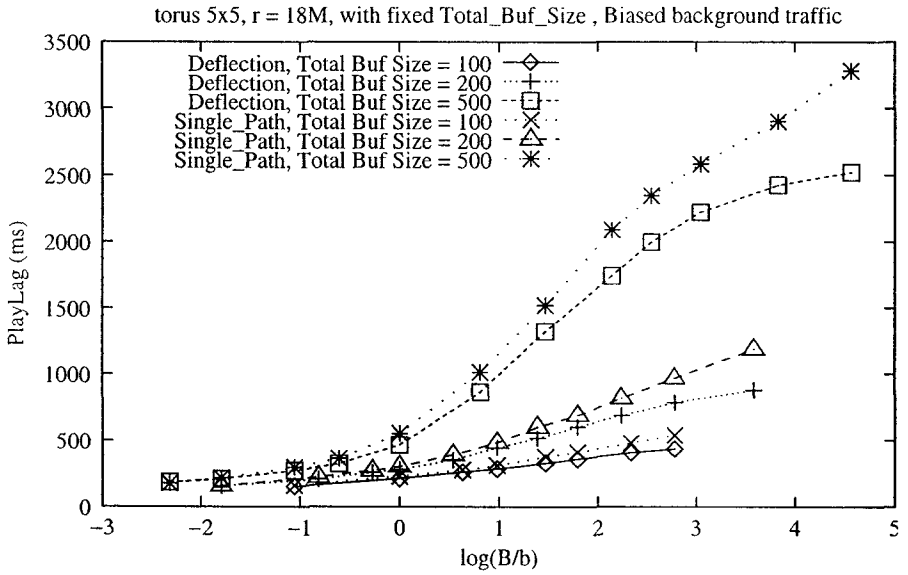


Figure 6.9: Playback Lag vs. log(B/b), Biased Load.

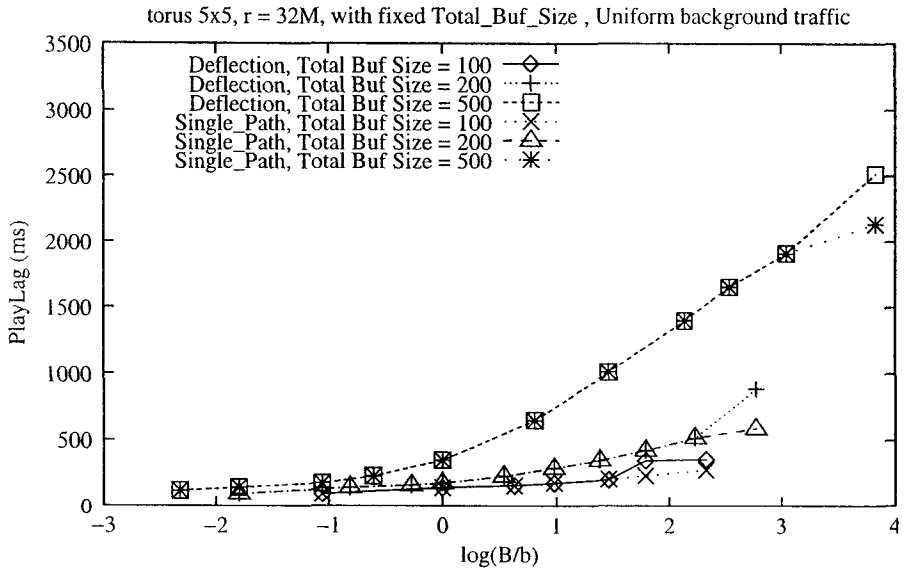


Figure 6.10: Playback Lag vs. log(B/b), Uniform Load.

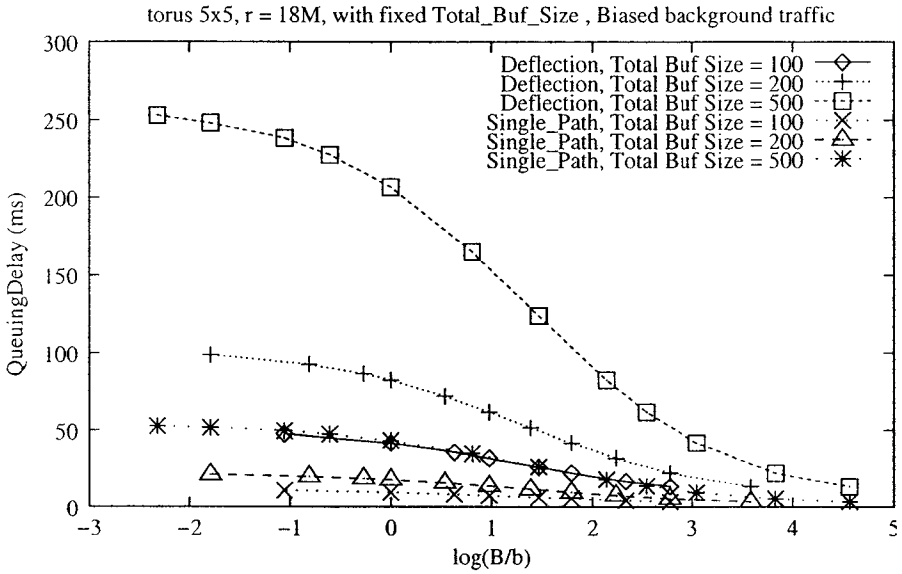


Figure 6.11: Queuing Delay vs. log(B/b), Biased Load.

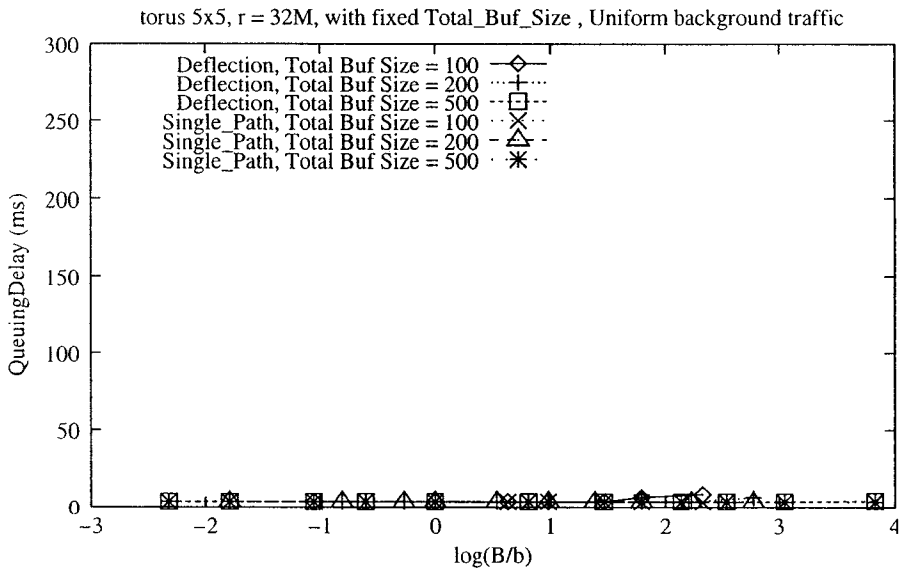


Figure 6.12: Queuing Delay vs. log(B/b), Uniform Load.

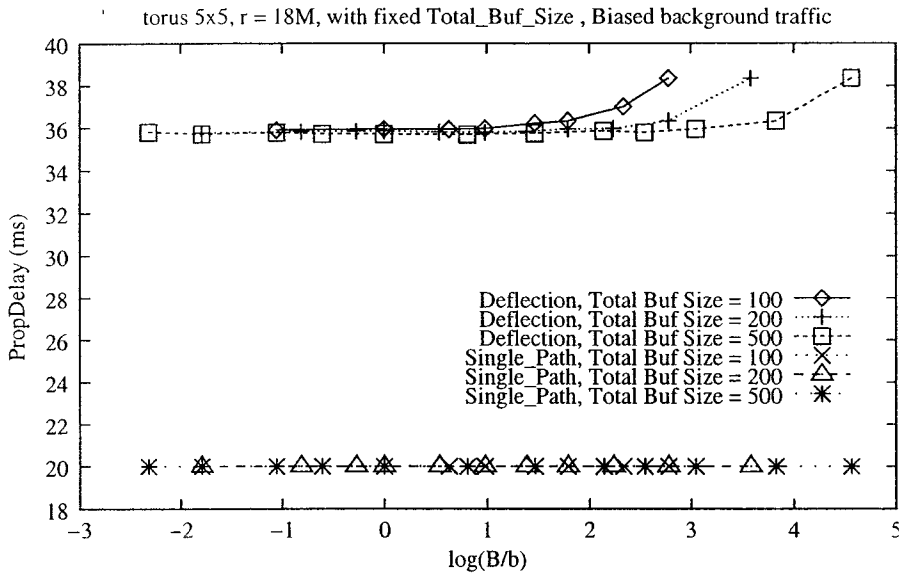


Figure 6.13: Propagation Delay vs. log(B/b), Biased Load.

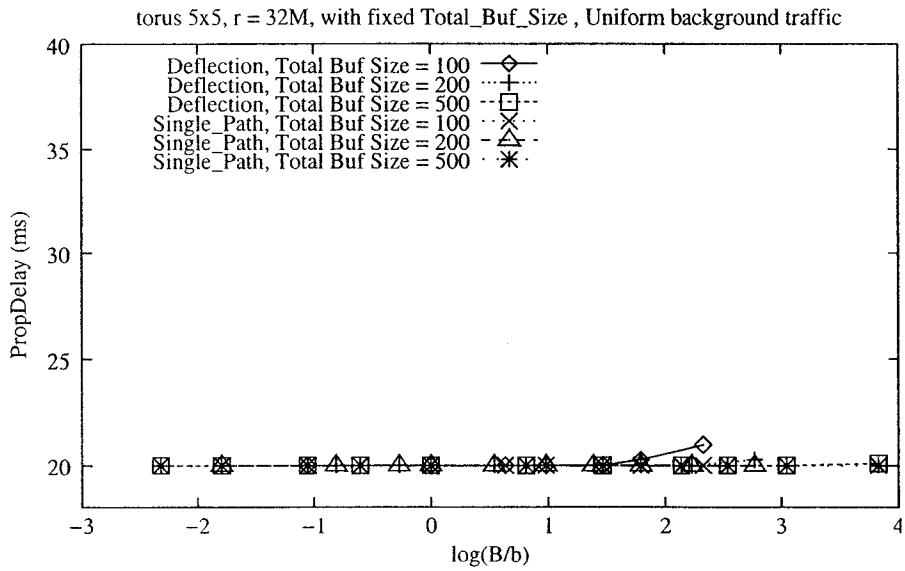


Figure 6.14: Propagation Delay vs. log(B/b), Uniform Load.

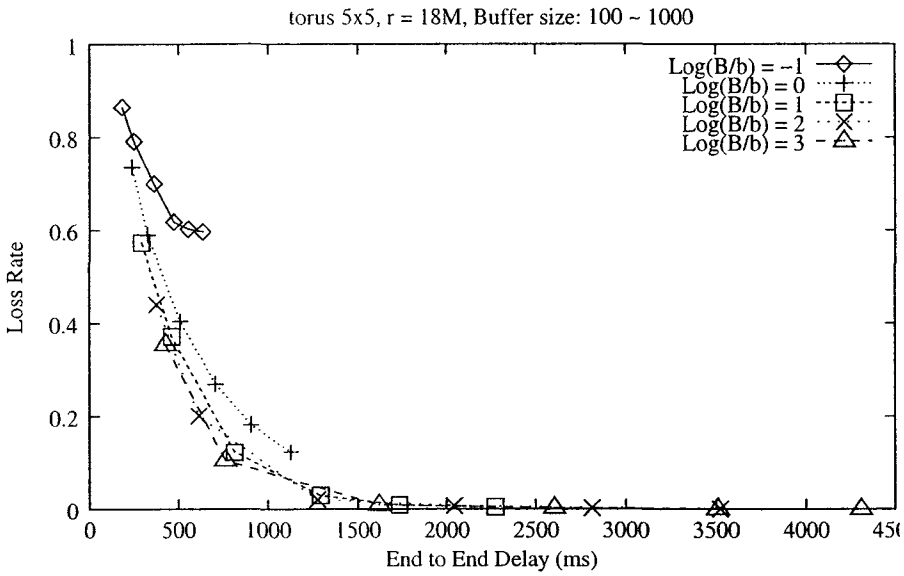


Figure 6.15: Torus 5x5, Loss rate vs. end–to–end delay, Biased Load.

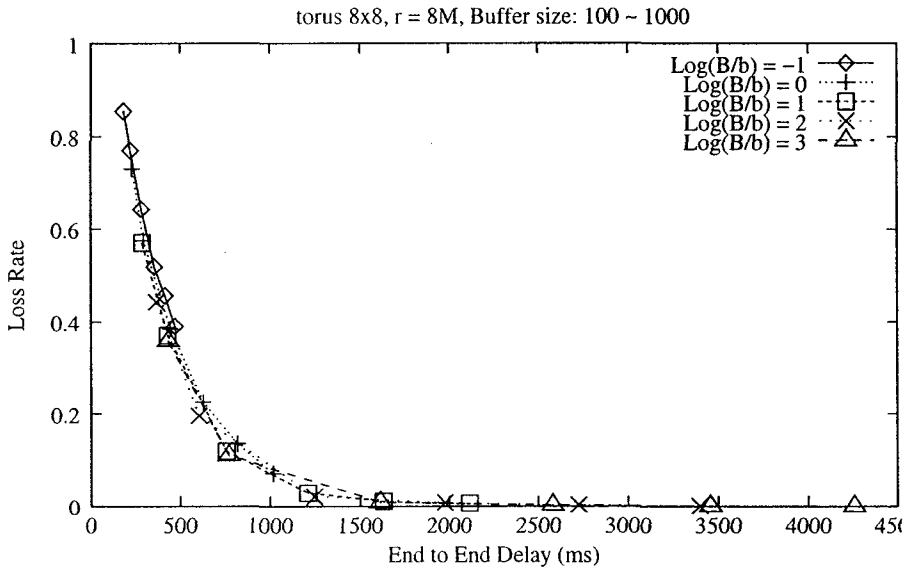


Figure 6.16: Torus 8x8, Loss rate vs. end–to–end delay, Biased Load.

Figures 6.15 and 6.16 show that these two performance measures can be traded to some extent. Each curve represents a single $B/b$ ratio, with the total amount of buffer space ranging from 100 to 1000 packets. We find that, for a wide range of the total amount of buffer space, some $B/b$ ratios (e.g., 0, 1, 2, and 3 in Figure 6.15), offer consistently acceptable loss and delay measures. It is also interesting to note that a large ratio of $B/b$ results in a better combined loss and delay performance. In other words, a large destination buffer combined with a small buffer at the router is more likely to provide both a satisfactory loss rate and an acceptable end–to–end delay.

The relatively small amount of buffer space at the routers at which they appear to perform satisfactorily, and the sharp increase in the drop rate when that small amount is reduced below a certain minimum, are consistent with the observations made in [54]. In that study, it is shown experimentally that a moderate amount of buffer space available to the routers tends to drastically improve the maximum throughput of a deflection network, bringing it quickly to a level comparable to that of a network with infinite buffers. In confrontation with our results, this seems to suggest that a single–path router with a large amount of buffer space is doubly misconfigured: it should be using little buffer storage while following alternative paths.

### 6.5.2 The Impact of Traffic Load

So far, all the experiments are conducted under a fixed traffic load for each topology. In the following, we will examine how the loss and end–to–end delay metrics vary with traffic intensity, and how the DF and SP perform in various load situations.

Figure 6.17 shows that when the background traffic load is lower than a certain threshold, i.e., 21Mbps, the DF scheme exhibits a lower and more stable loss rate than the SP scheme. The superiority of the DF over the SP within a certain load range is consistent with our intuition: under biased traffic load, although the shortest path is congested, chances are good that other parts of the network are underutilized, and therefore deflection helps alleviate congestion by diverting traffic from congested areas to lightly loaded

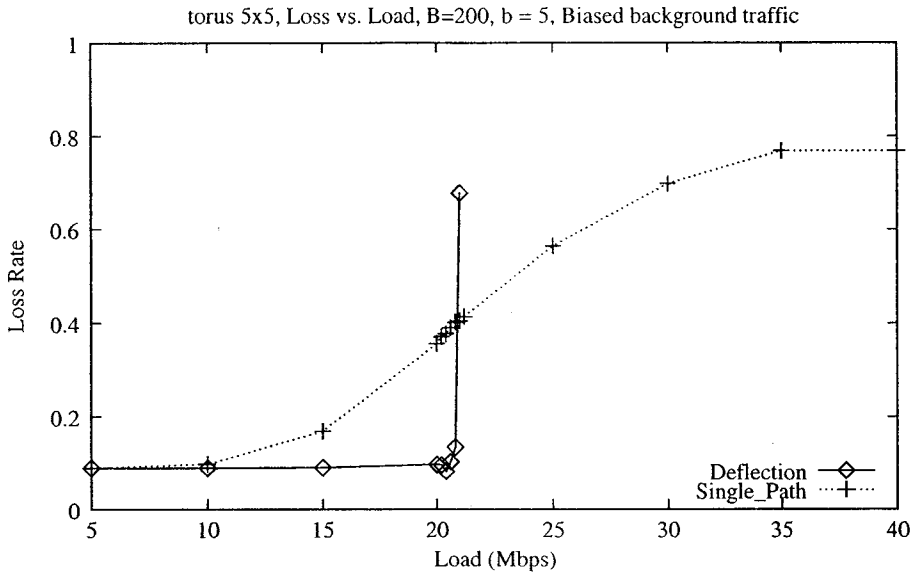


Figure 6.17: Loss Rate vs. Load, Biased Load.

areas; in contrast, confronted with the congestion on the shortest path, the SP scheme drops packets regardless of the congestion situation of the remaining part of the network.

Once the threshold is surpassed, the loss rate of the DF scheme soars high, because the network has been saturated with heavy traffic and deflection does not help any more; the SP curve rises more gradually because packet losses alleviate congestion to a certain degree, but its loss rate finally converges at a high value, i.e., almost 80%. Despite the better performance of the SP after the load threshold point, we find that after that point, the loss rates of the SP are too high to be usable. Therefore, we argue that deflection routing is the winner under biased traffic conditions.

For the uniform traffic situation, as shown in Figure 6.18, we find that the behavior of the DF curve resembles the one in Figure 6.17. But for the SP scheme, the loss rate curve increases much slower than in Figure 6.17. It is reasonable because for the same traffic intensity on the network, biased distribution causes heavy load on the shortest path while uniform distribution results in low load on it. In the former case, the DF can successfully relieve congestion while the SP is subjected to packet losses at the routers; in the latter case, the DF performs as well as the SP for a certain load range, but performs

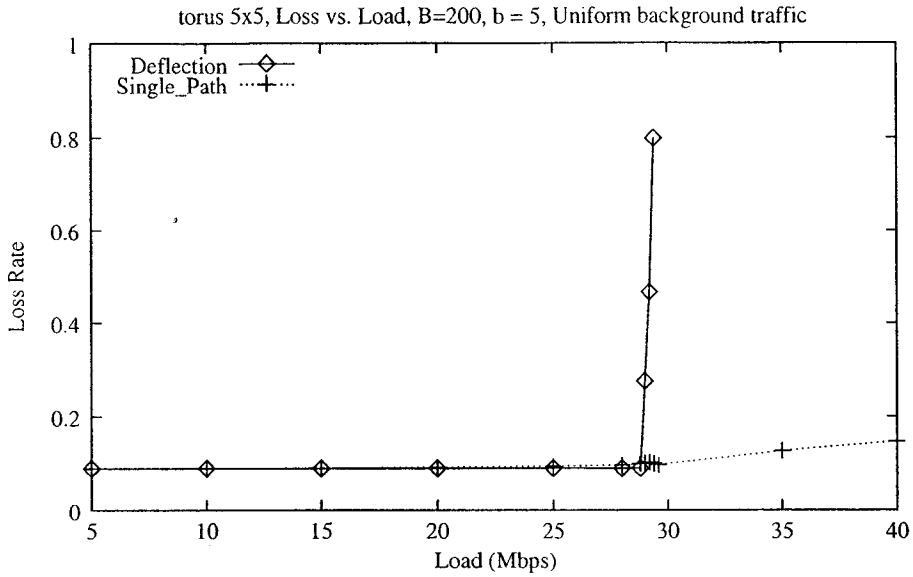


Figure 6.18: Loss Rate vs. Load, Uniform Load.

worse when load is higher. Considering that the biased traffic scenarios are more realistic than the uniform ones, we claim that deflection routing is a superior choice for realistic traffic scenarios.

Figure 6.19 and 6.20 demonstrate the end-to-end delay as a function of traffic load. Similar to above, we can observe the superiority of the DF over the SP when the network load does not surpass a certain threshold. Figure 6.21 and 6.22 shows the jitter trend. Intuitively, deflection routing causes higher jitter than single path routing, and the intuition is confirmed by these two graphs.

## 6.6 Conclusions

Our results suggest that deflection as a routing concept is less harmful than it would seem at first sight. From the global point of view of the entire network, the reassembly buffer is not a serious problem (and does not represent a new resource requirement) because its introduction reduces the resource requirements at the routers. Besides, the destination, being well aware of the specifics of its session, should be able to make better use of the reassembly buffer than a router having to cope with multiple and essentially unknown streams of traffic.

One standard argument against deflection networks is that the alternative

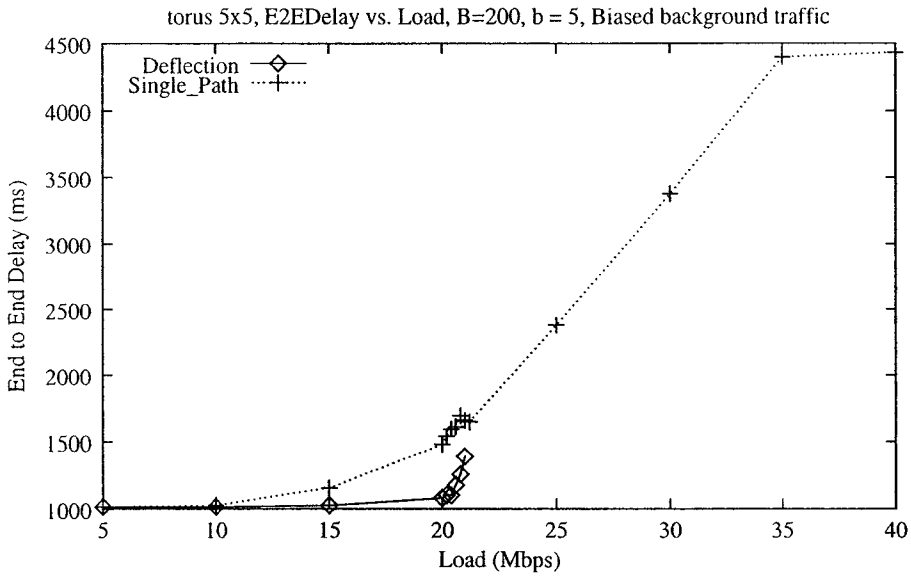


Figure 6.19: End–to–end Delay vs. Load, Biased Load.

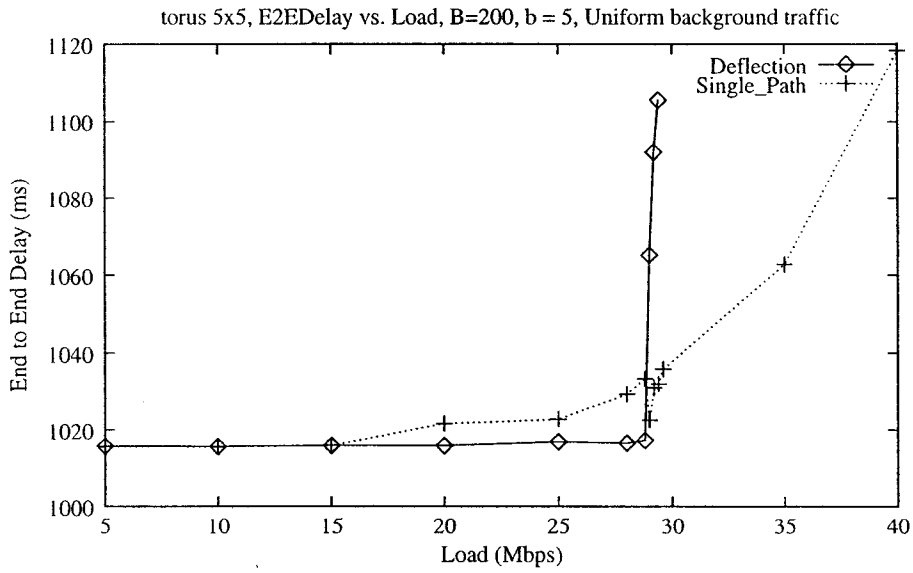


Figure 6.20: End–to–end Delay vs. Load, Uniform Load.

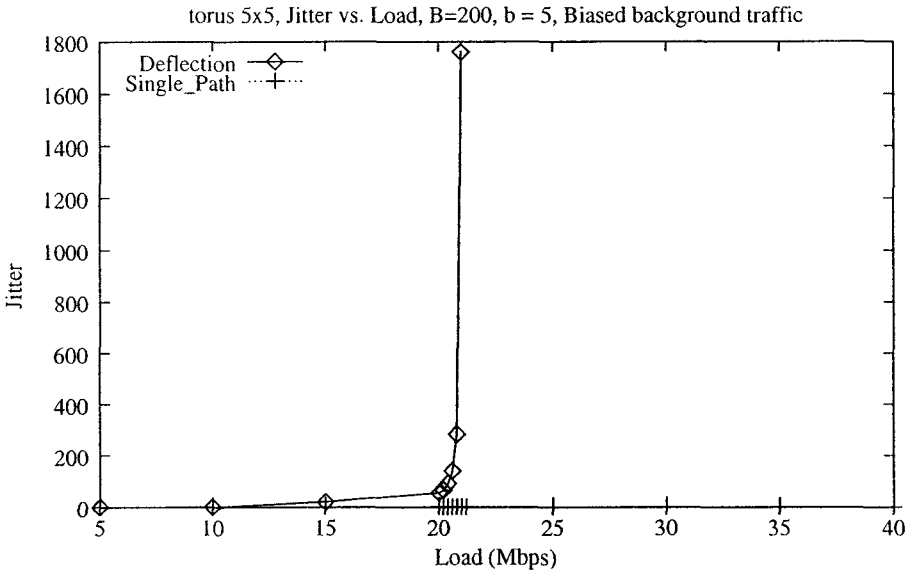


Figure 6.21: Jitter vs. Load, Biased Load.

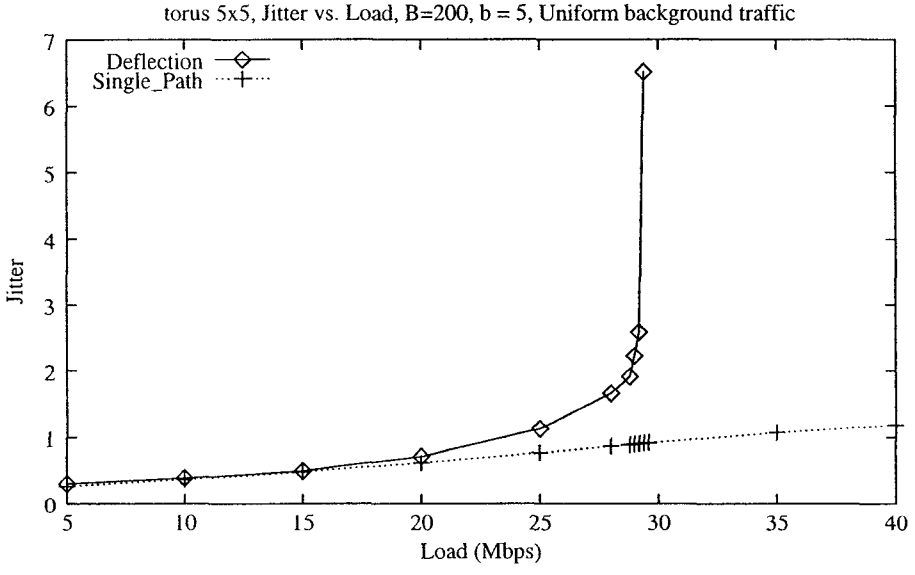


Figure 6.22: Jitter vs. Load, Uniform Load.

routes incur excessive jitter, which has a detrimental impact on the performance of isochronous sessions. Note, however, that by buffering packets that cannot be forwarded immediately, store–and–forward networks hardly solve this problem. While a deflection network can lose packets that fall outside the window provided by the reassembly buffer, a store–and–forward network can drop packets because of the lack of storage at the routers, or because those packets have been delayed too much to be useful. There is no fundamental difference at this level.

The issue of packet reassembly is often misguided (and brought forward as an erroneous argument against multiple–path routing) because of the insistence of some legacy applications on viewing their sessions as ordered sequences of packets. If we look carefully at those communication scenarios that truly require the preservation of packet ordering, we see that they fit into three categories:

- Sessions that could be carried out with packets arriving in any order (e.g., file transfers); they enforce packet ordering because the applications have been (unnecessarily) designed that way.

- Sessions involving relatively short transfers (e.g., a piece of text to appear on the screen), which can be reassembled in a trivially small buffer space.

- Long sustained isochronous streams (e.g., voice, video), which typically accept a non–zero packet loss and thus can be reasonably reconstructed within limited–size reassembly buffers. Note that in this category, the store–and–forward single–path approach doesn't guarantee zero packet loss either.

Let us consider the Internet, and TCP in particular. Due to the increasing transmission rates of links, most hosts today are capable of handling TCP sessions with relatively large bandwidth $\times$ delay products, which translate into large *advertised* TCP *receiver windows*. The receiver window plays the role of a large reassembly buffer capable of (a) re–ordering packets potentially arriving out of order, and (b) holding packets beyond "gaps" caused by losses, while

waiting for retransmissions to fill those gaps. However, the effectiveness of a large reassembly buffer to facilitate (b) is at least debatable, mostly because the dynamics of TCP constrict the window after a loss (and TCP's approach to window adjustments is quite conservative in general). Thus, case (b) provides reduced performance dividends but is typical in IP–based networks, where traditional single path routing is used. We argue that TCP would gain the full potential of receiver reassembly buffers if the balance was tilted in favor of case (a), which can be accomplished by deflecting packets when congestion occurs, instead of dropping them.

More experiments are required to verify the above speculations and put them on a more formal ground. Specifically, we need a better insight into the observed phenomena that would let us extrapolate them onto realistic irregular topologies and non–uniform traffic patterns. This is a natural direction for our further study.

# Chapter 7

# Conclusions and Future Work

## 7.1 Conclusions

With the rolling out of new network applications such as video conferencing, electronic media and Internet telephony, Quality of Service has become an important issue. New service models such as the Integrated Services and the Differentiated Services have been proposed to provide performance assurance and service differentiation. However, the underlying routing mechanism is still a "best-effort" one — all packets are routed undifferentiatedly using the shortest paths. This results in a poor match for the proposed QoS models, and therefore, to provide end–to–end QoS, reduce congestion, and optimize network-wide resource usage, QoS support is desired at the routing module.

One of the most important issues in QoS routing is scalability because the dynamic fluctuation of QoS status brings about enormously increased routing information exchanges. Simply reducing the frequency of routing updates is not a satisfactory solution because doing so causes inaccurate routing information and hence performance degradation. In addition, reducing routing information exchange frequency does not change the redundant and blind feature of flooding.

In this dissertation, we approached the scalability problem from three angles:

First, to compensate for routing inaccuracy, we proposed a multi–path construction algorithm and a set of path selection schemes, among which we found that the hop–based algorithms provide the best performance with imprecise

routing information. Compared with the single–path routing and other popular methods, our proposed WKS scheme shows superior capabilities in dealing with inaccurate information. Also examining the costs and benefits demonstrated that the proposed multi–path routing schemes are scalable solutions.

Second, based on the proposed multi–path mechanism, we attempted to reduce the link state information from two viewpoints. The first is the scope and frequency of link state updates. Different update priorities are assigned to nodes based on their locations. The proposed schemes need only minor modifications to the OSPF flooding mechanism while improving the cost–effectiveness of flooding. The second point of view is the content of the link state update messages. By qualitatively separating link state updates, we observed a noticeable performance gap, which implies that the quality of link state dissemination can be improved by distributing "good updates" more frequently.

Finally, deviating from the traditional connection–based QoS provisioning model, we re–approached the multi–path routing problem without depending on fixed path sets. Under the deflection model, we investigated the impact of multi–path routing on real–time delay jitter sensitive voice traffic. Using multiple paths for packets inside a session brings about the issue of packet mis–ordering in the network, and therefore assembly buffers are needed at the destination; on the other hand, the lossless feature of deflection routing allows us to reduce the buffer space at the intermediate routers. Consequently, we addressed the trade–off of buffer resource allocation between intermediate routers and destination hosts. We also conducted a comparison between the proposed lossless deflection strategy and the conventional shortest path strategy. Our results show that, from the global viewpoint of network resource management, deflection–based routing is a preferable choice.

## 7.2 Future Work

### 7.2.1 Deflection Routing vs. TCP

In Chapter 6, our experiments show that using deflection routing, we can improve the network performance by moving the buffer resources from the intermediate routers to the destination hosts, which reduces the costs of buffers and buffer management at the routers. We argue that the deflection routing, used in the TCP context, can make better use of the large bandwidth–delay products than the loss–based single–path routing. More specifically, a large bandwidth–delay product implies a large *advertised receiver window* size at the destination, which, according to TCP, encourages a source to send more traffic. This actually also accelerates the incurrence of network congestion. As a result, packets losses happen in the intermediate routers, which triggers the multiplicative reduction of the *congestion window* size at the source. Therefore, under TCP's loss–sensitive *conservative additive–increase–multiplicative–decrease* [19] window adjustment mechanisms, the loss–based traditional single path routing impedes the large destination buffers (or bandwidth–delay products) being fully used. In contrast, deflection would eliminate the possibility of packet loss and enhance the efficient usage of destination buffers. In the near future, we will conduct detailed experiments to verify our argument.

Also, how to adapt the TCP congestion control mechanism to the deflection network is another research problem in the future. For example, in deflection routing, packets are likely to reach their destinations out of order. This kind of packet mis–ordering is not necessarily caused by congestion, as assumed in TCP; consequently some modifications need to be incorporated into TCP to appropriately identify and respond to congestion in the deflection network. For another example, the re–transmission mechanism in TCP need re–consideration under the deflection assumption. Specifically, in the single–path loss–based routing paradigm, re–transmission is necessary because packets are likely to be lost. However, in the lossless deflection routing paradigm, when re–transmission timer times out, the packets which are presumed to be lost are still in the network; therefore, re–transmissions might actually aggravate

congestion.

### 7.2.2 Routing Issues in the Wireless Environment

Recent years have witnessed a tremendous growth in the use of mobile wireless devices, and routing issues in the wireless environment have been a hot research area, for example, routing in the *Mobile Ad hoc NETworks* (MANET) [76] and routing in the sensor networks [35]. A mobile ad hoc network (MANET) consists of a collection of mobile nodes, all of which may act as a router and dynamically create a wireless network among themselves without using any infrastructure or centralized administration (e.g., base stations). A sensor network is composed of small devices (or sensor nodes) with sensing, data processing, and communication capabilities. Sensor networks is a subset of MANETs, but its focus is to perform distributed sensing function for various applications, such as military surveillance, healthy monitoring, smart buildings, and seismic data acquisition, etc.

The wireless and self–organizing features make MANETs very useful in areas such as the military, emergency, instant conferencing, and "wearable" computing. However, high error rate and low bandwidth of channels and limited battery power pose new challenges to routing. For example, the routing scalability issue is very difficult in MANETs because the scarce bandwidth and battery power limit the size of a functional MANET and place more stringent requirements on the costs of routing information exchange in a MANET than in a wireline network. The scalability problem is even more pronounced in a sensor network environment, which might have tens of thousands of sensor nodes.

Also, QoS (routing) in a wireless environment is a very difficult problem due to the inherent problems of dynamic topology and unpredictable radio links, etc. It is still under debate whether or not QoS is necessary or possible to be provided in such a special environment. For example, what QoS exactly means in a sensor network environment and how to evaluate it are some interesting questions which have not been sufficiently explored.

Besides, security is a more acute issue in a wireless environment than in a

wireline environment. The MANET is more vulnerable to malicious attacks because of having a dynamically changing topology and lacking centralized monitoring and a clear line of defense [76]. It is easy for attackers to snoop and redirect network traffic, reply transmissions, and manipulate packet headers, etc. How to improve the security of the MANET from the viewpoint of routing protocols is an interesting research problem.

# Bibliography


[1] G. Apostolopolous. *Cost and Performance Trade-Offs of Quality of Service Routing.* PhD thesis, Computer Science, University of Maryland, College Park, August 1999.

[2] G. Apostolopoulos, R. Guerin, and S. Kamat. Implementation and Performance Measurements of QoS Routing Extensions to OSPF. In *Proceedings of IEEE INFOCOM'99*, March 1999.

[3] G. Apostolopoulos, R. Guerin, S. Kamat, T. Przygienda A. Orda, and D. Williams. QoS Routing Mechanisms and OSPF Extensions. IETF RFC 2676, December 1998.

[4] G. Apostolopoulos, R. Guerin, S. Kamat, A. Orda, and S. K. Tripathi. Intra-Domain QoS Routing in IP Networks: A Feasibility and Cost/Benefit Analysis. *IEEE Network*, September/October 1999.

[5] G. Apostolopoulos, R. Guerin, S. Kamat, and S. K. Tripathi. Improving QoS Routing Performance Under Inaccurate Link State Information. In *Proceedings of 16th International Teletraffic Congress (ITC-16)*, June 1999.

[6] G. Apostolopoulos, R. Guerin, and S. K. Tripathi. Quality of Service Routing: A Performance Perspective. In *Proceedings of ACM SIGCOMM'98*, September 1998.

[7] D. Awduche. IETF MPLS working group mailing list, http://cell.onecall.net/mhonarc/mpls/1998-Dec/msg00199.html, 1998.

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

[9] D. O. Awduche, A.Chiu, A. Elwalid, Indra Widjaja, and Xipeng Xiao. A framework for Internet Traffic Engineering. Internet Draft draft-ietf-tewg-framework-00.txt, Janurary 2000.

[10] Y. Bernet, P. Ford, R. Yavatkar, F. Baker, L. Zhang M. Speer, R. Braden, B. Davie, J. Wroclawski, and E. Felstaine. A Framework for Integrated Services Operation over Diffserv Networks. IETF RFC 2998, November 2000.

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

[12] V. A. Bolotin. Modeling Calling Holding Time Distributions for CCS Network Design and Performance Analysis. *IEEE JSAC*, 12(3):433–438, April 1994.

[13] F. Borgonovo and L. Fratta. Deflection Networks: Architectures for Metropolitan and Wide Area Networks. *Computer Networks and ISDN Systems*, 24:171–183, 1992.

[14] J.Y. Le Boudec and T. Przygienda. A Route Precomputation Algorithm for Integarate Service Networks. *Journal of Network and System management*, 3(4):427–449, 1995.

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

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

[17] L. Breslau, D. Estrin, D. Zappala, and L. Zhang. Limited Distribution Updates to Reduce Overhead in Adaptive Internetwork Routing. February 1993.

[18] W. Dobosiewicz C. Baransel and P. Gburzynski. Routing in Multi-hop Switching Networks: Gbps Challenges. *IEEE Network Magazine*, 3:38–61, 1995.

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

[20] A. Choudhury and N. Maxemchuk. Effect of a finite reassembly buffer on the performance of deflection routing. In *Proceedings of the International Conference on Communication(ICC)*, volume 3, pages 1637–1646, 1991.

[21] I. Cidon, R. Rom, and Y. Shavitt. Multi-Path Routing Combined with Resource Reservation. In *Proceedings of IEEE INFOCOM'97*, pages 92–100. IEEE, April 1997.

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

[23] B. Davie and Y. Rekhter. *MPLS Technology and Applications.* Morgan Kaufmann Publishers, 2002.

[24] D. Estrin, Y. Rekhter, and S. Hotz. A unified approach to inter-domain routing. IETF RFC 1322, May 1992.

[25] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On Power-law Relationships of the Internet Topology. In *Proceedings of ACM SIGCOMM 1999*, Aug 1999.

[26] The ATM Forum. Private Network-Network Interface specification version 1.0. Technical Report af-pnni-0055.000, ATM Forum, March 1996.

[27] The ATM Forum. Traffic Management Specification Version 4.0. Technical Report af-tm-0056.000, ATM Forum, April 1996.

[28] P. Gburzynski and J. Maitan. Deflection Routing in Regular MNA Topologies. *Journal of High Speed Networks*, 2(2):99–131, 1993.

[29] R. Guerin. IETF MPLS working group mailing list, http://cell.onecall.net/mhonarc/mpls/1998-Dec/msg00231.html, 1998.

[30] R. Guerin and A. Orda. QoS-based Routing in Networks with Inaccurate Information: Theory and Algorithms. *IEEE/ACM Transaction on Networking*, 7(3):350–364, June 1999.

[31] F. Hao and E. W. Zegura. Scalability Techniques in QoS Routing. Technical Report GIT-CC-99-16, College of Computing, Georgia Institute of Technology, 1999.

[32] F. Hao and E. W. Zegura. On Scalable QoS Routing: Performance Evaluation of Topology Aggregation. In *Proceedings of IEEE INFOCOM'00*, March 2000.

[33] C. Hedrick. Routing Information Protocol. IETF RFC 1058, June 1988.

[34] G. Huston. Next Steps for the IP QoS Architecture. IETF RFC 2990, November 2000.

[35] C Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCom'2000)*, August 2000.

[36] J. Jaffe. Algorithms for Finding Paths with Multiple Constraints. *Networks*, 14:95–116, 1984.

[37] R. Jain. *The Art of Computer Systems Performance Analysis.* John Wiley & Sons, Inc., 1991.

[38] Y. Jia, I. Nikolaidis, and P. Gburzynski. Multiple Path Routing in Networks with Inaccurate Link State Information. In *Proceedings of IEEE International Conference on Communications (ICC'2001)*, pages 2583–2587, June 2001.

[39] Y. Jia, I. Nikolaidis, and P. Gburzynski. Alternative Paths vs. Inaccurate Link State Information in Realistic Network. In *Proceedings of International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2002)*, pages 162–169, July 2002.

[40] Y. Jia, I. Nikolaidis, and P. Gburzynski. On the effectiveness of alternative paths in qos routing. *submitted to the International Journal on Communication Systems*, July 2002.

[41] Y. Jia, I. Nikolaidis, and P. Gburzynski. Qualitative Link State Dissemination Control in QoS Routing. In *Proceedings of the 3rd International Conference on Internet Computing (IC 2002)*, pages 140–147, June 2002.

[42] Y. Jia, I. Nikolaidis, and P. Gburzynski. Buffer Space Tradeoffs in Multihop Networks. In *Proceedings of the 1st Annual Conference on Communication Networks and Services Research (CNSR 2003)*, pages 74–79, May 2003.

[43] A. Juttner, B. Szviatovszki, I. Mecs, and Z. Rajko. Lagrange relaxation based method for the qos routing problem. In *Proceedings of IEEE INFOCOM 2001*, pages 782–791, April 2001.

[44] F. P. Kelly. Routing and Capacity Allocation in Networks with Trunk Reservation. *Mathematics of Operations Research*, 15:771–793, 1990.

[45] T. Korkmaz and M. Krunz. Multi-constrained optimal path selection. In *Proceedings of IEEE INFOCOM 2001*, pages 834–843, April 2001.

[46] W. C. Lee. Topology aggregation for hierarchical routing in ATM networks. *ACM Computer Communication Review*, 25(2), April 1995.

[47] Q. Ma. *Quality-of Service Routing in Integrated Services Networks.* PhD thesis, Carnegie Mellon University, January 1998.

[48] Q. Ma and P. SteenKiste. On Path Selection for Traffic with Bandwidth Guarantees. In *Proceedings of IEEE International Conference on Network Protocols (ICNP)*, October 1997.

[49] Q. Ma and P. Steenkiste. Quality-of-Service Routing for Traffic with Performance Guarantees. In *Proceedings of IFIP Fifth International Workshop on Quality of Service*, pages 115–126, May 1997.

[50] D. Magoni and J. J. Pansiot. Comparative Study of Internet-like Topology Generators. Technical report, LSIIT laboratory, Universite Louis Pasteur, May 2001.

[51] G. Malkin. RIP Version 2, Carrying Additional Information . IETF RFC 1388, January 1993.

[52] E.Q.V. Martins, M.M.B. Pascoal, and J.L.E. Santos. The K Shortest Paths Problem. Technical report, CISUC, June 1998.

[53] N. Maxemchuk. The manhattan street network. In *Proceedings of IEEE GLOBECOM'85*, pages 255–261, 1985.

[54] N. Maxemchuk. Comparison of deflection and store-and-forward techniques in manhattan-street network and shuffle-exchange networks. In *Proceedings of IEEE INFOCOM'89*, pages 800–809, April 1989.

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

[56] S. Nelakuditi, Z. Zhang, and R. P. Tsang. Adaptive proportional routing: A localized qos routing approach. In *Proceedings of IEEE INFOCOM 2000*, pages 1566–1575, March 2000.

[57] H. D. Neve and P. V. Mieghem. TAMCRA: A Tunable Accuracy Multiple Constraints Routing Algorithm. *Computer Communications*, 23:667–679, 2000.

[58] W. Olesinski and P. Gburzynski. Real-time traffic in deflection networks. In *Proceedings of WMC'98: Communication Networks and Distributed Systems, Modeling and Simulations*, pages 23–28, January 1998.

[59] W. Olesinski and P. Gburzynski. Service guarantees in deflection networks. In *Proceedings of the Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems(MASCOTS'01)*, pages 267–274, August 2001.

[60] M. Peyravian and A.D. Kshemkalyani. Network Path Caching: Issues, Algorithms, and a Simulation Study. *Computer Communications*, 20:605–614, 1997.

[61] A. Orda R. A. Guerin and D. Williams. QoS Routing Mechanisms and OSPF Extensions. In *Proceedings of 2nd Global Internet Miniconference (joint with Globecom'97)*, November 1997.

[62] K. Nahrstedt S. Chen. An Overview of QoS Routing for the Next Generation High-speed Networks: Problems and Solutions. *IEEE Network*, pages 64–79, November/December 1998.

[63] K. Nahrstedt S. Chen. Distributed QoS Routing with Imprecise State Information. In *Proceedings of the 7th International Conference on Computer Communications and Networks (ICCCN '98)*, October 1998.

[64] K. Nahrstedt S. Chen. On Finding Multi-constrained Paths (MCP). *The International Journal of Computational Geometry and Applications*, pages 874–879, 1998.

[65] A. Shaikh, J. Rexford, and K. Shin. Efficient Precomputaion of Quality of Service Routes. Technical report, July 1998.

[66] A. Shaikh, J. Rexford, and K. G. Shin. Dynamics of Quality-of-Service Routing with Inaccurate Link-State Information. Technical Report CSE-TR-350-97, Dept. of Electrical Engineering and Computer Science, University of Michigan, November, 1997.

[67] S. Shenker and C. Partridge. Specification of Guaranteed Quality of Service. IETF RFC 2212, September 1997.

[68] B. Uyless. *ATM: Foundation for Broadband Networks.* Prentice-Hall, Inc., 1995.

[69] Z. Wang. *Internet QoS, Architectures and Mechanisms for Quality of Service.* Morgan Kaufmann Publisher, 2001.

[70] Z. Wang and J. Crowcroft. QoS routing for Supporting resource reservation. *IEEE Journal of Selected Areas of Communications*, September 1996.

[71] J. Wroclawski. Specification of of the Controlled-Load Network Element Service. IETF RFC 2211, September 1997.

[72] X. Yuan and X. Liu. Heuristic algorithms for multi-constrained quality of service routing. In *Proceedings of IEEE INFOCOM 2001*, pages 355–364, April 2001.

[73] X. Yuan and A. Saifee. Path selection methods for localized quality of service routing. In *Proceedings of the 10th International Conference on Computer Communications and Networks (ICCCN 2001)*, October 2001.

[74] X. Yuan, W. Zheng, and S. Ding. A comparative study of qos routing algorithms that tolerate imprecise state information. In *Proceedings of the 11th IEEE International Conference on Computer Communications and Networks (ICCCN'02)*, October 2002.

[75] H. Zhang. Service Disciplines For Guaranteed Performance Service in Packet-Switching Networks. In *Proceedings of the IEEE*, volume 83, pages 1374–1396, October 1995.

[76] Y. Zhang and W. Lee. Intrusion detection in wireless ad-hoc networks. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCom'2000)*, pages 275–283, August 2000.

[77] Z. Zhang, C. Sanchez, B. Salkewicz, and E. Crawley. QoS Extensions to OSPF, work in progress.