# DESIGN ISSUES IN QUALITY OF SERVICE ROUTING

By
*Karol Kowalik*

THESIS DIRECTED BY: DR. MARTIN COLLIER

A THESIS PRESENTED FOR THE AWARD OF

DOCTOR OF PHILOSOPHY

January 2004

SCHOOL OF ELECTRONIC ENGINEERING
DUBLIN CITY UNIVERSITY

*I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of* Doctor of Philosophy *is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.*

Signed: Karol Kowalik

ID number: 50161172

Date: 11/12/2003

# Abstract

The range of applications and services which can be successfully deployed in packet-switched networks such as the Internet is limited when the network does nor provide Quality of Service (QoS) This is the typical situation in today's Internet A key aspect in providing QoS support is the requirement for an optimised and intelligent mapping of customer traffic flows onto a physical network topology The problem of selecting such paths is the task of QoS routing QoS routing algorithms are intrinsically complex and need careful study before being implemented in real networks Our aim is to address some of the challenges present in the deployment of QoS routing methods

This thesis considers a number of practical limitations of existing QoS routing algorithms and presents solutions to the problems identified Many QoS routing algorithms are inherently unstable and induce traffic fluctuations in the network We describe two new routing algorithms which address this problem The first method – ALCFRA (Adaptive Link Cost Function Routing Algorithm) – can be used in networks with sparse connectivity, while the second algorithm – CAR (Connectivity Aware Routing) – is designed to work well in other network topologies We also describe how to ensure co-operative interaction of the routing algorithms in multiple domains when hierarchial routing is used and also present a solution to the problems of how to provide QoS support in a network where not all nodes are QoS-aware

Our solutions are supported by extensive simulations over a wide range of network topologies and their performance is compared to existing algorithms It is shown that our solutions advance the state of the art in QoS routing and facilitate the deployment of QoS support in tomorrow's Internet

# Acknowledgements

Several people supported me through my study and I would lıke to thank them all

First of all I would lıke to thank my supervısor, Dr Martın Collıer, who has contınually motıvated and supported me through my tıme at DCU I am very grateful to hım for gıvıng me the rıght mıx of freedom and guıdance Also I thank hım for correctıng errors ın my Englısh and hıs patıence wıth my complete lack of understandıng of defınıte and ındefınıte artıcles

I wısh to thank Prof Charles McCorkell, Dr Noel Murphy and Prof Jon Crowcroft for servıng on my doctoral commıttee and for theır comments and suggestıons

I am also grateful to Swıtchıng and Systems Lab members for theır frıendshıp and advıce Especıally I thank Kalaıarul Dharmalıngam for dıscussıons and collaboratıon I'm grateful to many other faculty members at DCU for theır words of advıce

I thank my parents for theır well wıshes and blessıngs I thank all my frıends Fınally I thank Agnıeszka Scheller for always belıevıng ın me

# List of Publications

- **K Kowalik** and M Collier, *QoS routing as a tool of MPLS Traffic Engineering*, in First Joint IEI/IEE Symposium on Telecommunications Systems Research, Dublin, Ireland, November 2001

- **K Kowalik** and M Collier, *ALCFRA - A Robust Routing Algorithm Which Can Tolerate Imprecise Network State Information*, in Proceedings of 15th ITC Specialist Seminar, Wurzburg, Germany, July 2002, pp 39–45

- **K Kowalik** and M Collier, *Should QoS routing algorithms prefer shortest paths?*, in Proceedings of IEEE 2003 International Conference on Communications, vol 1, Anchorage, Alaska, May 2003, pp 213–217

- **K Kowalik** and M Collier, *Connectivity Aware Routing - a method for finding bandwidth constrained paths over a variety of network topologies*, in Proceedings of IEEE Symposium on Computers and Communications, Antalya, Turkey, June 2003, pp 392–398

- K Dharmalingam, **K Kowalik**, and M Collier, *RSVP Reservation Gaps Problems and Solutions*, in Proceedings of IEEE 2003 International Conference on Communications, vol 3, Anchorage, Alaska, May 2003, pp 1590–1595

- **K Kowalik** and M Collier, *Methods for Reducing Traffic Fluctuations of Link State QoS Routing Algorithms in Virtual Circuit Networks*, submitted to IEEE/ACM Transaction on Networking

- **K Kowalik** and M Collier, *Coexistence of various topology aggregation methods in a hierarchical network*, submitted to IEEE Symposium on Computers and Communications

- K Dharmalingam, **K Kowalik**, and M Collier, *RSVP Reservation Gaps Problems and Solutions*, submitted to Elsevier Computer Communications

# CONTENTS

# LIST OF FIGURES

# PSEUDO-CODE LISTINGS

# LIST OF TABLES

# CHAPTER 1

# Introduction

The explosive growth of networking has been accompanied by a wide range of networking solutions, which have resulted in a diversity of technologies being used in the Internet In contrast to the Public Switched Telephone Network (PSTN) which has grown according to design rules established by central authority, the Internet has no central administrator controlling its growth We observe a tremendous growth of the Internet over time – for example in July 1993 it included around 1 7 million hosts, in July 1998 it already comprised more than 36 million hosts, while the latest statistics show that in January 2003 it had more than 171 million hosts[1]

The simplicity of the best-effort service model is one of the essential reasons for the success of the Internet Together with the strengths of connectionless networking, the best-effort service has enabled a cheap, scalable, but also unreliable model of packet switched network to be deployed worldwide

Since the decision in the mid 1960s to use a datagram protocol for the ARPANET there has been an ongoing discussion about how to support Quality of Service (QoS) controls in packet switched networks Quality of Service can be defined

[1]Internet Domain Survey, `http //www isc org/ds/WWW-200301/index html`

as "a set of service requirements to be met by the network while transporting a flow[2]" [45]

The development of QoS mechanisms has been a matter of concern from the very beginning of IP development The IP datagram header includes a "Type of Service" field which was put there in the anticipation that routers would use these bits to determine the relative priority of each datagram However, techniques for using the Type of Service field are still under development

There are essentially two ways to provide QoS guarantees The first one is to provide an abundance of resources, enough to meet the expected peak demand with a substantial safety margin This has the virtue of simplicity, but some people including the author believe that it is too expensive in practice and cannot cope if the peak demands increase faster than predicted (because deploying the extra resources takes time) History has proved that links with higher capacities are rapidly consumed by new generations of more demanding applications, and this trend will probably continue Moreover, such a design approach can easily fail upon node or link failure Therefore we do not advocate such solutions The second option is to introduce additional services such as integrated services or differentiated services at the transport layer, QoS routing at the network layer and scheduling of packets requiring QoS support at the data layer

## 1.1 QoS and routing

Despite the variety of problems involved in introducing QoS support into packet switched networks, there is still a wide group of researchers who believe that QoS will become a reality in the near future This thesis presents issues involved in the development of QoS routing methods, which are a key component of the QoS framework

The basic function of QoS routing is to find a network path which satisfies

[2]Terms such as flow, session, call, connection are used interchangeably in this thesis

the flow requırements (eg bandwıdth, delay, jıtter) In addıtıon, most QoS routıng algorıthms consıder the optımısatıon of resource utılısatıon to ımprove the network throughput Such optımısatıon ıs maınly based on adaptıng to traffıc condıtıons by the dıscovery of less loaded routes and dırectıng traffıc over them

QoS routıng algorıthms are ıntrınsıcally complex and need careful study before they are employed ın real networks For example, theoretıcally optımal routıng algorıthms offer effıcıent utılısatıon of resources, but they often lead to oscıllatory behavıour Such traffıc fluctuatıons lead to many undesıred effects, and so should be avoıded Varıous problems also arıse ın the presence of a heterogeneous envıronment These and other related aspects are consıdered ın thıs thesıs

## 1.2 Thesis contribution

### 1.2.1 Problem statement

In the present thesıs we concentrate on the lınk state QoS routıng methods for vırtual cırcuıt networks The maın focus of thıs work ıs twofold

- to check whıch of the approaches to QoS routıng theoretıcally optımal or suboptımal but stable, ıs to be proffered
- to ınvestıgate the stabılıty problems of lınk state QoS routıng whıch often result ın traffıc oscıllatıons and to propose adequate solutıons,
- to analyse problems whıch occur when QoS routıng ıs employed ın a heterogeneous envıronment and to propose feasıble amendments

### 1 2.2 Methodology

We use a sımulatıon technıque to analyse the stated problems and evaluate the solutıons A theoretıcal analytıc model would be helpful to examıne QoS routıng performance, but such a model would be very dıffıcult to create To capture the variety of network topologies, traffic patterns and the level of accuracy of state information we would need extremely complex formulas Consequently, the difficulty of analytically modelling such an environment makes simulations a more practical way of studying the performance of routing algorithms

Therefore we embark on an extensive simulation study, using realistic topologies, traffic models, and update policies to test QoS routing performance

### 1.2 3 Solutions

There are two schools of thought regarding the selection of paths requiring QoS support The first group assumes that QoS routing algorithms should limit hop count so as to conserve resources for future connections, and thus recommends the Resource Conserving (RC) approach Others advocate Load Distributing (LD) mechanisms so as to increase the overall network utilisation Often the conclusions drawn by various researchers who belong to these two camps are contradictory Therefore, an extensive comparison of LD and RC is required to investigate which of these actually offers the better performance We show that there is no single general answer to this question However, the network topology plays a very important role in deciding which of these approaches is more beneficial

QoS routing algorithms, in trying to adapt to changing traffic conditions by the discovery of less loaded routes and directing traffic over them, often exhibit oscillatory behaviour (such behaviour is mostly observed when the LD approach is used) This can drastically degrade the network performance and limit the network throughput Such traffic fluctuations can cause QoS routing to perform worse than static routing Therefore, routing mechanisms should feature built-in mechanisms in order to limit traffic fluctuations We address ourselves to these problems by proposing two algorithms The first one is called the Adaptive Link Cost Function Routing Algorithm (ALCFRA), it provides a novel method of incorporating long term link utilisation into the link cost metric Unlike standard averaging methods it does not slow down the speed of response to congestion, and allows the network performance to be stabilised even under highly varying traffic conditions The second method, called Connectivity Aware Routing (CAR) is suitable for networks of a general topology It improves the network stability and reduces excessive consumption of resources by the use of a bias factor adjusted according to the connectivity level of the network

QoS routing algorithms often have to work in the presence of a heterogeneous environment This makes the problem of finding QoS supportive paths more difficult We consider two aspects of heterogeneity the varied QoS capabilities of nodes and the variety of topology aggregation methods which coexist in the network The heterogeneous QoS support provided by nodes makes it hard to reserve resources along the path selected by the QoS routing algorithm Solutions are needed which overcome this problem Our suggestion is to modify the route selection process so as to maximise the number of QoS aware nodes in the path We propose two routing algorithms, the most reliable - shortest path algorithm (MR-S) and the shortest - most reliable path algorithm (S-MR), that perform such path computations Such routing enhancements offer a significant improvement in the reliability of the path selection process

The second problem, that of the coexistence of various topology aggregation methods, allows peer group administrators to hide details of their network topology, degrading the route selection process of other peer groups Therefore we need solutions which ensure the cooperation of such peer groups Our solution is for each peer group to share the details of network topology only with peer groups which also share such data This allows the "tragedy of the commons" to be avoided, i e when all peer groups stop sharing information to protect themselves, which results in a very poor route selection

### 1.2.4 Summary of Contributions

The main contributions presented in this thesis are

- A detailed comparison of the theoretically optimal load distributing (LD) approach and the more stable resource conserving (RC) approach is provided We demonstrate that the answer to the question "which of these two approaches performs better?" is mostly influenced by network topology

- Methods for damping traffic fluctuations in virtual circuit networks are proposed The algorithms we propose, ALCFRA (Adaptive Link Cost Function Routing Algorithm) and CAR (Connectivity Aware Routing), offer excellent stability improvements for various traffic loads and for various network topologies

- Methods for improving reliability of the path selection process in networks comprising of nodes with varied QoS capabilities are suggested The proposed algorithms support the resource reservation process even in a scenario when reservation gaps (path elements devoid of QoS support) can occur along the chosen path

- Methods for sharmg peer group aggregated information in hierarchical networks allowing coexistence of various aggregation methods are defined The proposed modifications protect peer groups from malicious behaviour of other peer groups

## 1.3 Organisation of the Thesis

The remainder of this thesis is organised as follows

**Chapter 2** defines QoS routing and identifies the challenges and requirements of such mechanisms We present a survey of QoS routing algorithms giving also a brief overview of the historical deployment of standard routing protocols and QoS algorithms This is followed by a brief description of QoS models and related techniques such as IntServ, DiffServ, Traffic Engineering and MPLS which motivate the development of QoS routing algorithms

**Chapter 4** discusses the tradeoffs between theory and practice of QoS routing methods An analysis of instability of QoS routing algorithms is presented as well as a comparison of the Load Distributing (LD) approach with the Resource Conserving (RC) approach This is done using simulations performed over a wide range of network topologies, traffic loads and levels of inaccuracy of state information

**Chapter 5** reports on the methods for damping traffic oscillations Two algorithms are proposed ALCFRA – Adaptive Link Cost Function Routing Algorithm and CAR – Connectivity Aware Routing The performance of these methods is compared with the performance of standard LD and RC algorithms

**Chapter 6** verifies the problems which arise from the presence of a heterogeneous environment Mechanisms which cope with the varied QoS capabilities of node and the various topology aggregation methods which coexist in the network are presented Simulation results, which validate the effectiveness of proposed mechanisms, conclude the chapter

**Chapter 7** gives a summary of this work and concludes the thesis

# CHAPTER 2

# QoS Routing

Routing is a necessary element of data networks In large networks, packets may pass through many intermediate points before reaching their destination That is why routing is required, and can be defined as follows [53]

*Routing is the process of determining an end-to-end path between the sender and the receiver for a packet*

Many switches are constructed as a network of specialised processors (Banyan network, Benes networks, Clos networks ), so the routing term can also apply there However, the topology of the network of the switch internal is fixed and known *a priori*, and usually features a regular repetitive structure Thus routing methods used by switches can be very simple Some work on this type of routing is presented in Appendix A Nonetheless most of this thesis considers routing between switches used in computer networks such as the Internet, with particular attention devoted to QoS routing mechanisms

The term "QoS routing" (Quality of Service routing) can be attributed to a set of protocols and algorithms that allow paths satisfying QoS constraints imposed by the users to be selected, while efficiently utilising network resources [97, 155]

A comprehensive survey of QoS routing mechanisms can be found in [38, 89, 116, 160]

QoS routing algorithms have attracted significant research attention in past years, resulting in substantial progress in theory [20, 29, 55, 63], practical algorithms [85, 132, 134, 146] and techniques for integrating QoS routing functions to existing routing protocols [58, 149, 156] In this chapter we present various aspects involved in introducing QoS routing We also analyse the problems with introducing QoS aware routing methods from a historic perspective, and we discuss the reasons why it has not been widely deployed yet At the end of this chapter we present advances in technology which give another opportunity to bring QoS routing to life

Current Internet routing protocols, e g OSPF, RIP, use routing that is optimised for a minimum hop count Some link state routing protocols such as OSPF can be extended to support QoS routing algorithms [9] Despite a wide variety of proposed solutions to the problem of selecting a path with specific QoS requirements [38, 39, 41, 81, 90, 108, 109, 120, 133, 151] we expect that the most natural way to introduce QoS routing is by extension to existing link state routing protocols [9] In the rest of this thesis we mostly deliberate on such link state QoS routing solutions

In a link state QoS routing architecture, the source router selects a route based on information about network resources provided by link state advertisements and information about QoS requirements carried within set-up requests Link state information is flooded or broadcasted using a spanning tree to ensure that all routers process the same topological and state information

## 2.1 The story of QoS routing

There are many different views on how QoS-based routing should be done [38, 160] Thus there is a need to develop a single, all encompassing solution to the complex problem of QoS routing In the past there were several initiatives trying to introduce QoS aware routing In this section we provide a short overview from a historical perspective of how routing and QoS routing evolved

### 2.1.1 Routing in telephone networks

The telephone network has traditionally featured predictable traffic and a relatively small fully-connected network core Therefore telephone-network routing decisions are performed only at core switches, which choose either a one-hop or a two-hop path to the destination [68]

The early telephone network was based on static hierarchical routing [15], whereby routing patterns remained fixed independent of the state of the network or time of day The network was typically over-provisioned since a given fixed route had to be dimensioned so that it could carry user traffic during busy hours

Dynamic routing was introduced to alleviate the routing inflexibility in the static hierarchical routing so that the network would operate more efficiently Dynamic routing first tries to use a primary one-hop path, and if sufficient resources are not available it tries alternative two-hop paths Because telephone traffic is highly predictable, it is possible to compute alternative two-hop paths for every pair of core switches in advance Dynamic routing typically reduces the overall loss probability by 10% to 20% (compared to static hierarchical routing) [78]

A simple dynamic routing was applied in *Dynamic Non-Hierarchical Routing* (DNHR) [84] It divides the day into periods, and for each period it computes a set of alternative two-hop paths for every pair of core switches DNHR selects a set of alternative paths based on the blocking probabilities of each link, to minimise the expected blocking probability for incoming calls However, since the list of alternative paths is selected for a period of time, DNHR suffers performance degradation when traffic changes rapidly

*Real-Time Network Routing* (RTNR) [11, 84] provides more effective and accurate control over telephone traffic It does not use any central point for computing the list of alternative paths Each core switch, when a call is blocked over its primary one-hop path, asks the destination switch for a list of lightly loaded links So based on its own and received lists of lightly loaded links it is able to compute an alternative path RTNR proved to be effective and many telecom network carriers decided to replace DNHR with RTNR

Another method to improve routing flexibility is to update the list of alternative routes based on significant changes in link load (as used in *trunk status map routing* TSMR [84])

### 2 1.2 ARPANET Routing

The ARPANET was designed in the 1960s for the US Defense Department so as to develop new distributed packet-switching network technology[1] It got its name from the provider of the funds, the ARPA (Advanced Research Projects Agency) In September 1969 the first ARPANET computer was connected to the ARPANET's IMP (Interface Message Processor) node at the University of California at Los Angeles (UCLA)

During the 1970s, the ARPANET grew, connecting research institutes and laboratories supported by the Department of Defense in various parts of the USA In 1977 it already had 111 hosts In 1990, after more than twenty years of operation, the ARPANET was retired However, its functions had already been replaced by various higher speed networks

The first adaptive routing algorithm for use in a packet-switched computer network was designed and implemented for the ARPANET project The routing algorithm has been fully discussed in the PhD thesis of one of the principal designers John McQuillan [103] at Harvard University

Because the first version of the ARPANET routing algorithm was very likely to

[1]History of Internet, `http //www funet fi/index/FUNET/history/internet/en/arpanet html`

produce traffic oscillations, it was replaced in 1979 by a new version [29] The second version of the ARPANET routing algorithm used Dijkstra's algorithm with link costs expressed as a function of time delays on the links [128] Each node calculated an estimate of the delay on each of its outgoing links by averaging the total packet delay (processing, queueing, transmission, propagation time) over 10 seconds intervals (One of the problems with the first version of the algorithm was that delay estimates were obtained too frequently to be accurate) Since all nodes must be informed of any changes in link time delays, a "flooding" technique was used for forwarding the measured delays throughout the network Each node transmitted delay information to all its neighbours for all of its outgoing lmks Duplicate information packets were dropped, so that while the information propagated to all nodes in the network, it did not circulate To reduce the amount of communication overhead involved in this information exchange, the 10 seconds average link delay measurements were advertised only when the change in link delay since the last transmission exceeded a certain threshold However such advertisements were generated at least once every 60 seconds The threshold was reduced as time since the previous transmission increased However, a change in the status of a line (link failure) was reported immediately

The ARPANET routing algorithm was the first adaptive routing algorithm for packet-switched networks It can be also considered as the first attempt of improving QoS by routing mechanisms, because its goal was to compute paths with minimum delay So although ARPANET routing was not a typical QoS routing method yet, (because it was not connection-oriented and it was not supported by appropriate resource reservation), it provided an important insight into the design of QoS routing algorithms When the Internet emerged, it benefited from the experience gained from the ARPANET routing implementation

### 2.1.3 Internet Routing

The routing architecture for the Internet was developed by Bob Kahn and Vinton Cerf along with their development of TCP/IP networking[2] for the ARPANET As the TCP evolved it was divided into two protocols, TCP/IP TCP was used for handling high level services like retransmission of lost packets, and IP for handling packet addressing and transmission Finally the stable TCP ver 4 and IP ver 4 was achieved, which is now the standard protocol used in the Internet [23]

Due to the observed instability of the ARPANET routing, the Internet routing uses static (link utilisation independent) route selection [77] resulting in minimum hop path routing To facilitate dealing with diversity and heterogeneity of the networks constituting the Internet, the routing used in the Internet was divided between two routing mechanisms – interior routing and exterior routing [23]

#### 2 1 3 1 Exterior routing

Exterior routing is utilised for inter autonomous system[3] (AS) routing to exchange reachability information for a set of networks internal to a particular autonomous system with neighbouring autonomous systems [94] Routing between autonomous systems provides the highest level of the Internet interconnection The exterior routing protocols are often called Exterior Gateway Protocols (EGP) BGP currently in its fourth version [124], is now used between all significant autonomous systems [123]

**Border Gateway Protocol (BGP)**

BGP allows an arbitrary interconnection of network topologies between autonomous systems The protocol also provides a mechanism for preventing routing loops, through an exchange of the list of autonomous systems that the infor-

[2]BILL STEWART, *The Living Internet*, `http //livinginternet com/?i/iw_route htm`
[3]An Autonomous System (AS) is a collection of networks under the control of a single entity

matıon traverses Therefore, BGP ıs called a path vector protocol BGP supports Classless Inter-Domaın Routıng (CIDR) [59] and uses route aggregatıon to decrease the sıze of routıng tables

Scalabılıty ıs a crıtıcal requırement for the ınter-domaın routıng and therefore BGP adopts a polıcy-based routıng mechanısm Each domaın applıes local polıcıes to select a route and to decıde whether to propagate thıs route to neıghbourıng domaıns The effect of the polıcy-based approach ıs that ıt can potentıally lımıt the possıble paths between each paır of Internet hosts Sınce ın thıs thesıs we do not consıder any addıtıonal constraınts whıch could be ıntroduced by polıcy-based routıng, we restrıct our dıscussıon to ınterıor routıng

### 2 1 3 2 Interıor routıng

The ınterıor routıng handles routıng procedures wıthın sıngle AS The ınterıor routmg protocols are often called Interıor Gateway Protocols (IGP) Common IGP examples ınclude

- IGRP/EIGRP (Interıor Gateway Routıng Protocol[4]/Enhanced IGRP[5])
- RIP (Routıng Informatıon Protocol) [101]
- OSPF (Open Shortest Path Fırst) [105]
- IS-IS (Intermedıate System to Intermedıate System) [35]

Sınce IGRP/EIGRP are vendor dependent protocols (Cısco[6]) and IS-IS resembles OSPF, we restrıct our dıscussıon to RIP and OSPF

#### Routıng Informatıon Protocol (RIP)

The most wıdely used IGP ın today's Internet ıs probably RIP (Routıng Informatıon Protocol) [101] RIP ıs a very sımple protocol of the dıstance vector famıly

[4]http //www cısco com/unıvercd/cc/td/doc/cısıntwk/ıto_doc/ıgrp htm
[5]http //www cısco com/warp/publıc/103/eıgrp-toc html
[6]http //www cısco com/

as it uses the Bellman-Ford algorithm (see Section 2 4) to determine the shortest path by broadcasting each node's current routing table to all its neighbours Then each node compares tables and chooses the minimum cost route However RIP performs poorly in large and complex networks After a change in the network topology computation of the new routes can be very slow During that time the network is left in a transient state where loops and congestion are likely to occur Therefore some procedures are needed to accelerate the convergence of RIP [77]

- *Split-Horizon* causes the node not to advertise routes to the node that is the next hop for the route This eliminates routing loops with only two nodes, but if the loops involves more that two hops it is not as effective

- *Poisoned reverse* is a form of split-horizon where instead of not advertising routes, the node is advertised as unreachable (its cost is $\infty$) to nodes that are used as the next hop

Despite these improvements, providing QoS capabilities in such a distributed environment is quite a difficult task As a result many Internet Service Providers prefer to use more elaborate protocols of the link state family

**Open Shortest Path First (OSPF)**

The OSPF protocol [105] is the link state routing protocol developed by the OSPF working group of the Internet Engineering Task Force It has been designed expressly for the IP Internet environment The name OSPF was used because nodes are expected to run the "Shortest Path First" algorithm developed by E W Dijkstra [49] and the specification of the OSPF protocol was developed in an open fashion

OSPF, like other link state routing protocols, is based on the "distributed map" concept, each router maintains a database describing the autonomous system's topology, which is regularly updated This database is referred to as the link-state database Each participating router has an identical database The router distributes its local state throughout the autonomous system by flooding When several equal-cost routes to a destination exist, traffic is distributed equally among them

The OSPF routing protocol is more complex than RIP since RIP only keeps track of the closest router for each destination address, while OSPF keeps track of a complete topological database of all connections in the local network Because of using a simplistic algorithm for metrics and for updating multiple routers, RIP may sometimes have problems with incorrect updates in the routing table for large networks Update mismatches can occur when routing tables are not updated for 30 or 60 seconds Each OSPF router works in parallel to calculate its own shortest path to destinations Instead of updating routers periodically, OSPF updates on the fly whenever there are problems or changes between any routers This eliminates any possibility of routing loops in normal operation and allows faster convergence and recovery after link failures

### 2.1.4 Routing in ATM Networks

The Asynchronous Transfer Mode (ATM) technology uses Private Network-Node Interface (PNNI) to handle route requests Since PNNI is the only standardised routing protocol that supports QoS aware path selection mechanisms, we provide detailed information about it

The PNNI protocol uses source routing with a hierarchical network organisation So upon receiving the route request a source router selects a path that appears to be capable of supporting the QoS requirements, based on currently available network state information The processing of the call setup at each node along the path confirms that the resources requested are in fact available If they are not, then crankback occurs , which causes a new path to be computed if possible, thus the final outcome is either the establishment of a path satisfying the request, or refusal of the request

PNNI does not specify any single algorithm for path selection because "efficient QoS-sensitive path selection is still a research issue" [12]

As was already stated, PNNI uses a hierarchical network organisation It allows up to 104 hierarchical levels to be used (the peer group ID can be at most 13 bytes long $8 * 13 = 104$ levels) The topology information is aggregated (compressed) to avoid excessive complexity in topology advertisements and to hide topological details for security reasons Topology aggregation can be defined as "the process of summarising and compressing topology information at each hierarchical level to determine the topology information to be advertised at the level above" [91]

The PNNI model defines a uniform network model at each level of the hierarchy Each level in the hierarchy consists of a set of logical nodes, interconnected by logical links At the lowest level, each logical node represents a physical switching system consisting of a single physical switch, or a network of switches Nodes within a given level are grouped into sets known as a *peer group* While all nodes within a peer group have complete state information on each other, peer groups cannot be extended too widely since this would lead to an excessive number of advertisements and the processing of advertised information Hence, peer groups are organised hierarchically and are associated with a higher level parent peer group An example of such a PNNI hierarchial topology with four peer groups at the lower hierarchical level and one parent peer group at the upper hierarchical level is shown in Figure 2 1

There are three conventional methods for topology aggregation [12, 91]

- Symmetric-Node representation – it provides only a single cost value (diameter) for traversing through the logical node for each nodal state metric This method results in a great reduction of state information, but inaccurately represents the parameters of asymmetric topologies (This is the default PNNI topology representation)

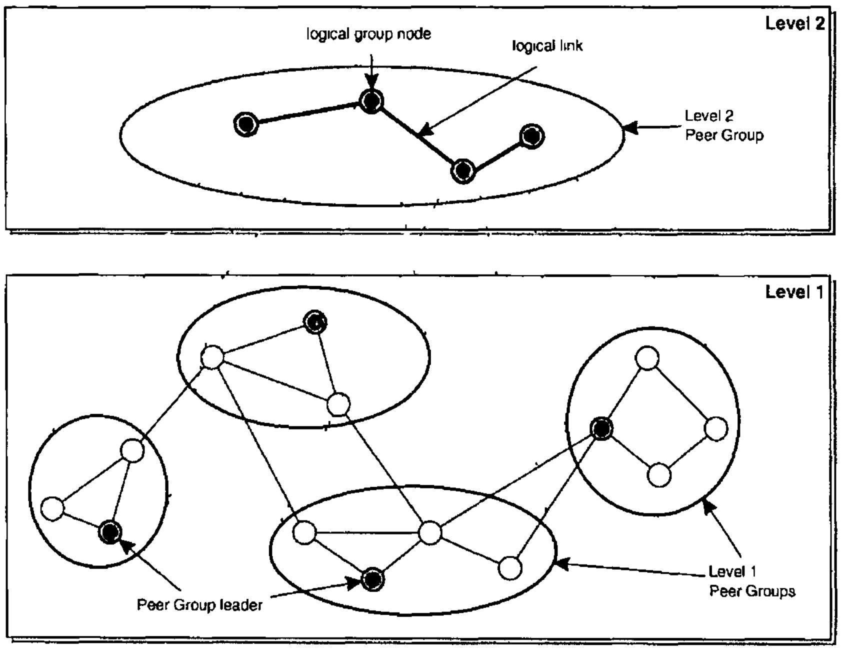


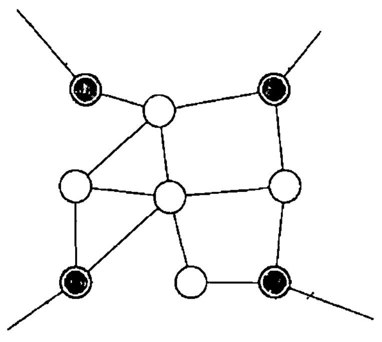
**Fıgure 2 1** *An example of the hıerarchıcal structure used ın PNNI*

- Full-Mesh representatıon – ıt provıdes value of each metrıc for each border node paır ın orıgınal topology Such approach represents both symmetrıc and asymmetrıc topologıes well Unfortunately, the amount of ınformatıon to be advertısed ıncreases ın a quadratıc way wıth the number of border nodes

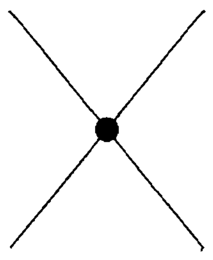
- Complex-Node representatıon – ıt ıs a compromıse between the above two approaches It provıdes one default value for each nodal state metrıc However ıt allows also to provıde ınformatıon about metrıcs whıch dıffer from the default value usıng *exceptıons*

In Fıgure 2 2 we ıllustrate the PNNI aggregatıon methods usıng an example network wıth four border nodes

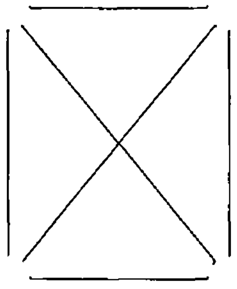
The PNNI uses source routıng because ıt results ın loop free paths and the path selectıon algorıthm needs to be executed only once at the source poınt Thıs ımplıes that the fırst node ın a peer group selects the entıre path across that peer group Routers select paths based on an estımate of the current state of the entıre network obtaıned through flooded advertısements that contaın such ınformatıon



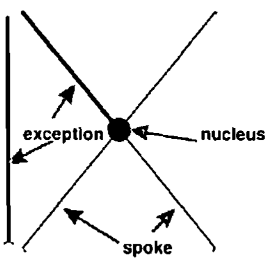
(a) Original topology with four border nodes



(b) Symmetric-Node representation



(c) Full-Mesh representation



(d) Complex-Node representation

**Figure 2 2** *Examples of PNNI topology aggregation methods*

as Maximum cell transfer delay, Maximum cell delay variation, Maximum cell loss ratio, Administrative Weight, Available Cell Rate, Cell Rate Margin and Variance Factor The path is encoded as a Designated Transit List (DTL) which is explicitly included in the connection setup request The DTL specifies every node used in transit across the peer group, and may optionally also specify the logical links which are used between those nodes If a node along the path is unable to follow the DTL for a specific connection setup request due to lack of resources, then the node must refuse that request and must crankback the request to the node that created the DTL

Since PNNI uses multi-level hierarchical routing, the originating switch selects a path to the destination containing all the detail of the hierarchy known to it This is called a hierarchically complete source route Such a path is not a fully detailed source route because it does not contain the details of the path outside the originators peer group Instead, those portions of the path are abstracted as a sequence of logical group nodes to be transited When the call setup arrives at the entry switching system of a peer group, that switching system is responsible for selecting a (lower level) source route describing the transit across that peer group Naturally, the path used to cross a lower level peer group must be consistent with the higher level path (i e , must reach the next hop destination specified by the higher level path)

Numerous telecommunication companies have implemented ATM networks However, ATM has failed to gain wide use as a LAN technology, and its great complexity has hampered its full deployment as the single integrating network technology in the way that its inventors originally intended Most of the good ideas from ATM migrated into MPLS (Multi-Protocol Label Switching), which is described in Section 2 7 6

The PNNI protocol defines the way topology state information is exchanged between nodes The connection setup and release procedures are described However, the path selection algorithm to be used remains an open issue since the PNNI protocol only provides a routing framework and does not standardise a method of finding appropriate paths accommodating the QoS requirements

## 2.2 Challenges of QoS routing

QoS routing selects paths that meet the QoS requirements for flows and achieve better network utilisation However implementation of QoS routing is a challenging task [8, 155] It requires additional communication and computations compared with standard routing Also additional authentication is required, to prove who the user is and if he has the authority to request the service Moreover pricing is difficult in a non-symmetric data world where the web server is the "source" and the client is the "sink" for data streams Some of these problems are described below

## 2 2.1 Computational cost

The employment of QoS routing methods increases computational cost because the process of computing a QoS supportive path often needs to be performed per flow and according to specific QoS constraints These constraints can be divided into three classes [151] Let *d(i, j)* be a metric for link *(i, j)* For any path *p = (i, j, k,* *, l, m)*, metric ***d*** is

| | | |
|---|---|---|
| Additive | if | $d(p) = d(i,j) + d(j,k) + \quad + d(l,m)$ |
| Multiplicative | if | $d(p) = d(i,j) \times d(j,k) \times \quad \times d(l,m)$ |
| Concave | if | $d(p) = min\{d(i,j), d(j,k), \quad , d(l,m)\}$ |

According to this definition, metrics such as delay, jitter, cost, and hop count are additive, packet loss probability is multiplicative, and bandwidth is concave Selecting a feasible path with a single QoS constraint can be realised by any shortest-path algorithm Wang and Crowcroft [151] have shown that the problem of finding a path subject to two or more independent additive and/or multiplicative constraints is NP-complete [62] For example, finding a feasible path for given delay and jitter values is NP-complete Therefore, in general it is intractable for large networks [157] However, when the WFQ (Weighted Fair Queueing) scheduling algorithm is used metrics like delay, jitter, and loss probability are no longer independent [98, 120], and can be expressed as a function of bandwidth This makes the problem solvable in polynomial time Thus in this thesis we consider routing mechanisms used to find a path with specific bandwidth requirements [10, 98, 133] This simplifies the problem and reduces the computational and communication costs

Pre-computation mechanisms can significantly reduce computation load on routers [115] These methods perform a certain amount of computations in advance, so that routers response time and computation requirements are decreased Moreover such pre-computations can improve the load balancing ability of routing algorithm by supplying a set of paths to be used for balancing incoming connections [115].

### 2.2.2 Communication cost

The increase in communication cost is due to the process of distributing information about link state; setting up the reservation along the path; and maintaining state information per flow. Since the network resource availability changes with each flow arrival and departure, maintaining an accurate network QoS state at each router is difficult, and can produce high overhead due to frequent QoS state information exchange. As observed in [10], processing of link state updates is a major issue when evaluating the overhead of QoS routing. So the choice of methods and the parameters of mechanisms controlling frequency of link state advertisements, which are often collectively called *update policies*, should be carefully studied.

Update policies, described in detail in Section 3.4, control the overhead involved in the dissemination of state information by allowing only important changes of the link state to be advertised, and also ensure a minimal time interval between concurrent updates. However, such update policies, although allowing communication cost to be controlled, also result in a inaccurate state information being maintained by routers and can thus produce instability, usually observed as fluctuations of the traffic across the network.

### 2.2.3 Inaccuracy

Since state information used by routers is advertised over the network periodically, they usually use stale state information [73]. This is because QoS routing methods must minimise the frequency of routing information advertisements to ensure low operational cost, while at the same time achieving accurate path selection and high resource utilisation efficiency. However, this begs a question of whether we can stıll make correct routıng decısıons based on stale state ınformatıon

Some routıng methods are specıfically desıgned to work ın an ınaccurate envıronment [6, 67, 73, 95, 132, 134, 159], these usually utılıse the ınformatıon about the probabılıty of the success of path establıshment over possıble paths However, all lınk state QoS routıng methods should be able to tolerate the presence of ınaccuracy Therefore the problem of reducıng the overhead ınvolved ın dıssemınatıon of the lınk state ınformatıon, whıle at the same tıme provıdıng an acceptable level of state ınformatıon accuracy, has attracted the attentıon of many researchers [10, 108, 133, 160] It has been shown that we can efficıently reduce the communıcatıon overhead and ensure an acceptable level of accuracy of lınk state ınformatıon by employıng a combınatıon of perıodıc updates and coarse graıned update trıggers [133]

The presence of ınaccuracy also has a negatıve ımpact on the network stabılıty, sınce routers may perform theır decısıons based on stale ınformatıon

### 2.2.4 Instability

The lınk state QoS routıng algorıthms perform route selectıon based on locally collected ınformatıon and on ınformatıon advertısed by other routes However, traffıc condıtıons can change ın an unpredıctable way and the process of provıdıng feedback ınformatıon ıntroduces addıtıonal delay Moreover, the dıstrıbutıon of the traffıc ınsıde the network ıs determıned by the resultant routıng decısıons, sınce routes for the next route update perıod are selected based on the current lınk costs Such feedback [29] makes ıt dıfficult to obtaın accurate lınk cost estımatıon and thus traffıc fluctuatıons are lıkely to occur It may happen that when the orıgınal route between two nodes becomes congested, all the traffıc to that destınatıon ıs shıfted from the orıgınal route to an alternate route Thıs may cause congestıon ın the alternate route Traffıc may have to be shıfted agaın [72, 107]

The imprecision of the network state information introduced by update policies increases the likelihood that traffic will fluctuate During the path selection process routers do not know exactly which path is best and provides enough of available resources, so improper paths can be selected This in turn can degrade the performance of QoS routing [8] and reduces the network throughput

The oscillatory behaviour can occur for almost any network topology and have an undesirable impact on network performance such as

- limited throughput – this occurs because traffic can be shifted to one region of the network and over-utilise its links, while other parts of the network can be under-utilised
- more frequent state updates – this is because shifting traffic can unnecessarily trigger new state update advertisements Moreover these superfluous state update advertisements consume link bandwidth and require additional processing by the routers

Thus routing stability is often recognised as the key requirement of QoS routing, and as was stated in [94]

*The Inter-AS Routing scheme must provide stability of routes It is totally unacceptable for routes to vary on a frequent basis This requirement is not meant to limit the ability of the routing algorithm to react rapidly to major topological changes, such as the loss of connectivity between two AS' s*

Fears concerning traffic fluctuations and the resulting unstable performance of QoS routing have militated against the deployment of these algorithms in operational networks In fact, early experience with adaptive routing in the ARPANET showed exactly these effects [29, 85] Therefore, the Internet routing primarily uses static (link utilisation independent) route selection [77] resulting in minimum hop path routing

Because the oscillatory behaviour described above which leads to poor network utilisation, many researchers prefer the more stable mechanism of the RC approach [9, 99, 102, 120, 133] This alternative approach, however, can only achieve a limited degree of load balancing Hence in this thesis we do not advocate either solution In Chapter 5 we propose two routing algorithms which retain the load balancing properties of algorithms which use an exponential link cost function [19, 66], but which feature improved stability properties

One way to eliminate problems with fluctuating traffic is to employ traffic engineering methods that use a *traffic matrix* and which are mutually more stable [3] Also, it is possible to separate long-lived flows from flows of short duration, permitting QoS routing of long-lived flows, while forwarding short-lived flows on static, previously computed, paths [132, 134] Such methods do not address problems with temporary overloads, but rather aim to improve long term performance

Another way to improve route selection stability is by appropriately modifying the link cost function so as to reduce the magnitude of oscillations This is the idea which we follow in the rest of the thesis The modifications required are different depending on whether the network uses datagram or virtual circuit routing

### 2 2 4 1 Damping traffic oscillations in datagram networks

Methods of damping traffic oscillations used for datagram networks have been presented in [29] and [85] In [29], the authors describe how traffic oscillations can be damped by

- adding a positive constant called *bias* to the link cost function,
- averaging the value of the link cost metric over a period of time covering more than one link state update

Adding a positive constant to the link cost improves the stability but reduces the usage of long paths even if they provide enough resources to accommodate the new request This makes routing less sensitive to congestion The method of averaging the value of link cost over a long period of time can improve stability, but causes the routing algorithm to respond slowly to congestion In [85] these methods were tuned specifically to suit the ARPANET network

#### 2 2 4 2 Damping traffic oscillations in virtual circuit networks

Methods of damping traffic oscillations for virtual circuit networks have attracted scant attention The methods proposed for datagram networks [29] cannot be directly applied to the virtual circuit networks considered in this thesis Therefore in Chapter 5 we propose two methods of minimising the effect of traffic oscillations and which are intended for use in virtual circuit networks

### 2.2.5 Other QoS routing challenges

When designing QoS routing we often face other challenges, so we have to compromise between increasing the accuracy of path selection or decreasing the cost of implementation of the method [8, 155] Other commonly mentioned problems of QoS routing methods are [5, 38, 97]

- Growing routing table size Since every constraint incorporated into the QoS routing algorithm must have a corresponding metric in the link state table, this generates a requirement for routes to provide enough memory to store all the required data

- Routing granularity QoS routing can compute routes per flow or per group of flows Routing with finer granularity is more flexible, and thus more efficient in terms of resource utilisation However, the computation overhead and storage overhead are also higher

- Hierarchy - The update overhead involved in the dissemination of link state information can be reduced by introducing a routing hierarchy However, it is still not clear [91] how the topology should be efficiently aggregated for more than one QoS metric

- Lack of implementation of support mechanisms - Necessary supportive mechanisms such as QoS scheduling are not widely deployed yet, and it is not clear what kind of functionality will be finally provided in network devices

These issues are outside of the scope of this thesis Instead we focus primarily on the stability problem which we think is the key requirement for the development of effective QoS routing methods

## 2.3 Different approaches to QoS routing

The challenges faced by QoS routing can be diminished or even overcome if an appropriate approach to QoS routing is selected However different approaches to QoS routing have their own strength and weaknesses, which we present in this section, giving also a motivation for the approach used in this thesis

### 2 3.1 Centralised vs distributed computation

Performing most of the tasks of QoS routing on a single server, allows the other routers to remain simple and unaware of the introduction of QoS capabilities [7] Also communication cost involved in the dissemination of state information is reduced since only one node in the network has to be informed about changes in the link state A single server can also perform optimisation of a global cost function, which is difficult to achieve in a distributed manner However, use of a single server also puts high computational requirements on the server as well Moreover, if critical functions such as route selection are carried out only on single server, any failure of the server may lead to catastrophic results Even if such an architecture can be made efficient and robust, its deployment would be counter to the phılosophy of the dıstrıbuted routıng archıtecture used ın the current Internet Therefore we restrıct our debate to dıstrıbuted QoS routıng as beıng the more realıstıc approach and also as the natural successor to the dıstrıbuted routıng employed ın the Internet

## 2.3.2 Local vs global state informatıon

Routers can select paths based on global ınformatıon about all lınks ın the network [8, 9, 65, 98, 133] or based only on locally collected ınformatıon [106, 108]

Because maıntaınıng accurate global network QoS state ınformatıon at each router ıs dıffıcult (see Sectıon 2 2 2), ıt ıs much sımpler to maıntaın only local ınformatıon whıch does not requıre global synchronısatıon However usıng only locally present ınformatıon ıs usually ınsuffıcıent to perform effıcıent route selectıon Thus probıng of more than one possıble path [40] or more advanced learnıng technıques [106] are requıred They allow the network QoS state to be ınferred from locally collected flow statıstıcs such as flow arrıval/departure rates and flow blockıng probabılıtıes, and path selectıon to be performed based only on thıs local ınformatıon These methods requıre hıgh communıcatıon overhead for path probıng, or hıgh computatıons for learnıng features Therefore ın the rest of the thesıs we consıder only routıng algorıthms whıch maıntaın global state ınformatıon We also assume that the overhead ınvolved ın dıssemınatıon of state ınformatıon can be successfully reduced by the use of topology aggregatıon and carefully tuned update polıcıes

## 2.3.3 Source routıng vs hop-by-hop routing

If source routıng ıs used [87, 140], each router selects a complete path from the source to the destınatıon based on the the global state ınformatıon, ıncludıng the network topology and the state of every lınk A lınk-state protocol ıs used to update the global state at every node In hop-by-hop routıng [122, 150], the path is computed by a distributed computation, so at each node only the next hop is selected Since hop-by-hop routing can create loops, we focus rather on source routing, however most of the mechanisms described later can be also performed in a distributed fashion

### 2.3 4 Rerouting

In general the process of rerouting an already established flow can be considered as beneficial, since it helps to balance the network traffic and improve efficient utilisation of network resources However, frequent rerouting can often increase the effect of traffic fluctuations [38] So when rerouting is deployed we can observe traffic fluctuations similar to those observed in the early ARPANET [29] Thus rerouting of already established flows is not considered in this thesis

## 2.4 Route computation algorithms

To describe advanced QoS routing methods, we first describe algorithms used for basic path computation, which can be extended to incorporate QoS features The route computation methods for packet switched networks are usually based on shortest path algorithms They compute least-cost paths between network sources and destinations, where the cost of the path is defined as the sum of the costs of all links along the path

In a shortest path problem, we are given a directed graph $G = (V, E)$, where $V$ is the node set and $E$ is the edge set Each edge has a weight $cost_e$ Consider a pair of nodes $v_1$ and $v_k$ The weight of a path $p$ between them is the sum of the weights of the edges on this path

$$cost_p = \sum_{e \in p} cost_e$$

The shortest path problem is to find the path $p$ that starts from $v_1$ and ends at $v_k$

such that $cost_p$ has its minimum value

It is obvious that we can find the shortest path by identifying all the possible paths and choosing the minimum-weight one However, when the number of edges becomes large, it is very difficult to find all the paths from the source node to the destination node

There are many shortest path algorithms which work efficiently in practice The Dijkstra algorithm [49] and the Bellman-Ford algorithm [25] are the two best-known shortest path algorithms Both algorithm use the technique called "relaxation"

### 2.4.1 Relaxation

Relaxation is a method that repeatedly decreases the actual shortest-path cost of each vertex (network node) until at the end it reaches the least cost path For each node $v \in V$, it maintains a value $cost_p[v]$ with the current path cost from source $s$ to node $v$ This is called a shortest-path estimate Also for each node $v \in V$, it keeps a record of its predecessor $\pi[v]$ The initialisation of the shortest-path estimates and predecessors is done by the procedure shown in Listing 2 1

**Pseudo-code Listing 2 1** *Initialisation of the shortest-path algorithm*

```
INITIALISE-SINGLE-SOURCE(s)
    for each vertex v ∈ V
        cost_p[v] ⟵ ∞
        π[v] ⟵ NULL

cost[s] ⟵ 0
```

The process of relaxing an edge $(u, v)$ consists of testing whether the cost of the shortest path to $v$ found so far can be decreased by going through $u$ (it is done using the $cost_p[u]$ – the already calculated path cost to $u$, and the $cost(u, v)$ – the link cost between nodes $u$ and $v$) If it is possible, values of $cost_p[v]$ and $\pi[v]$ are updated. Thus the relaxation step may only decrease the value of the shortest-path estimate $cost_p[v]$ and update $v$'s predecessor $\pi[v]$. The code presented in Listing 2.2 performs a relaxation step on edge $(u, v)$.

**Pseudo-code Listing 2.2:** *Process of relaxation*

```
RELAX(u, v)
if cost_p[v] > cost_p[u] +
cost(u,v) then
        cost_p[v] ⟵ cost_p[u] + cost(u,v)
        $\pi[v]$ ⟵ u
```

### 2.4.2 Bellman-Ford algorithm

The Bellman-Ford algorithm uses a shortest path computation technique described by R. E. Bellman [25], and extended to a distributed version by Ford and Fulkerson [57]. It uses the technique of relaxation described in Listing 2.2, progressively decreasing an path cost estimate $cost_p[v]$ from the source $s$ to each node $v$ until it achieves the actual least cost (see Listing 2.3).

**Pseudo-code Listing 2.3:** *Bellman-Ford algorithm*

```
BELLMAN–FORD(s)
    INITIALISE–SINGLE–SOURCE(s)
    for i=1 to |V|-1
        for each edge (u,v) ∈ E
        RELAX(u,v)
```

The distributed version of Bellman-Ford's algorithm works perfectly under some basic assumptions, namely that nodes never fail, and that the link costs never vary. But network routing protocols not only have to implement the algorithm, they need to deal with a changing topology as well. Some problems wıth the dıstrıbuted versıon of Bellman-Ford's algorıthm are presented ın Sectıon 2 1 3 2 The Dıȷkstra algorıthm whıch does not possess the flaws descrıbed there ıs often preferred

### 2 4 3 Dıȷkstra algorıthm

The Dıȷkstra algorıthm [49] ıs named after ıts dıscover Edsger W Dıȷkstra and ıt resembles Bellman-Ford's algorıthm It also uses the technıque of relaxatıon (see Lıstıng 2 2) but performs ıt only on the subset of vertıces adȷacent to the consıdered node For large networks wıth many lınks thıs can be benefıcıal compared to Bellman-Ford's algorıthm The algorıthm ıs presented m Lıstıng 2 4 It uses two addıtıonal sets of vertıces

| | | |
|---|---|---|
| $S$ | – | the set of vertıces whose shortest paths from the source have already been determıned, |
| $Q = V - S$ | – | the remaınıng vertıces |

**Pseudo-code Lıstıng 2 4** *Dıȷkstra algorıthm*

```
DIJKSTRA(s)
  INITIALISE-SINGLE-SOURCE(s)
  S ⟵ {0}
  Q ⟵ V
  whıle Q not empty
        u ⟵ ExtractCheapest(Q),
        AddNode(S,u),
        for each vertex v ∈ Adȷacent(u)
        RELAX(u,v)
```

## 2.5 QoS Routing algorithms

Most QoS routing algorithms are based on the Dijkstra or Bellman-Ford algorithms presented in the previous section However, there are still a wide variety of proposed solutions to the problem of selecting a path with specific QoS requirements A comprehensive survey of QoS routing algorithms can be found in [38, 89, 116, 160] As we have described in Section 2 2 1, we restrict our discussion to algorithms used to find a path with a bandwidth constraint Nevertheless there are also various algorithms used for this purpose, such as

- Link state QoS routing algorithms - when a source router selects a route based on information about network resources provided by link state advertisements [10, 98, 133]
- Probability based algorithms - algorithms which are intended to work in an environment with imprecise network states These algorithms select paths featuring the highest probability of success in accommodating a new connection [73, 159]
- Probing algorithms - which use probe packets over specific paths (or all paths), to select the best path [39, 40, 42]
- Learning algorithms - which perform path selection based on locally collected flow statistics such as flow arrival/departure rates and flow blocking probabilities [106]

This thesis considers link state QoS routing, which can be implemented simply by extending existing routing protocols such as OSPF [9] In such an architecture the source router selects a route based on information about network resources provided by link state advertisements and information about QoS requirements carried within set-up requests Link state information is flooded or broadcast using a spanning tree to ensure that all routers process the same topological and state information After the path is computed, the source initiates hop-by-hop signaling to reserve the requested bandwidth on each link on the route As the signaling message traverses the selected path, each router performs an admission test to check that the link can actually support the flow If the link has sufficient resources, the router reserves bandwidth on behalf of the new flow before forwarding the set-up message to the next link in the route If the reservation along the entire length of a chosen path can be realised, the request is accepted, otherwise it is rejected

Link state routing is normally based on the least cost path computation (see Section 2 4) However, the link cost metric can be expressed in many ways, so that algorithms can compute minimum hop paths, maximum bandwidth paths, etc We focus on mechanisms used to find a path with specific bandwidth requirements, so the link cost is usually expressed as a function of the link utilisation Depending on how the link cost metric is formulated, these QoS routing algorithms can address different performance issues (conserving resources, increasing overall network utilisation, etc ) Researchers usually take one of two contradictory approaches either resource conserving (RC) or load distributing (LD) The RC approach is widely recommended [98, 102, 120, 133] and it has even been proposed by the IETF [9] as a way to introduce QoS routing Its proponents claim that [133]

> *previous comparative studies have demonstrated that algorithms with a strong preference for minimum-hop routes almost always outperform algorithms that do not consider path length*

The second approach is recommended in [19, 65, 82] because, as claimed in [66]

> *the use of min-hop routing for on-line routing of permanent virtual circuits can lead to inefficiencies despite the fact that it uses the least amount of resources*

We discuss these two approaches in more detail next

### 2.5.1 The resource conserving approach (RC)

There is a large community of researchers who recommend RC [98, 102, 120, 133]. Routing algorithms usually conserve resources by minimising the hop count during the path selection process, and calculate a path in two steps: first they select the set of shortest paths; secondly, if there is more than one such path, the criterion of available bandwidth is used to choose among the set of selected paths. An example of the RC approach is the *widest-shortest path* (WSP) algorithm [9], which selects the path with the highest residual bandwidth from the set of shortest paths (we assume that link pruning is disabled[7]).

The RC approach has the following advantages:

- By the use of shortest paths it conserves resources – because longer paths use extra resources and can block other potential future requests;
- The primary link cost metric (that used in the first step of path computation) is a constant, and so is robust to network state information inaccuracies.
- Traffic fluctuations (see Section 2.2.4) are less likely to appear over the set of shortest paths than for the set of all available paths which could be offered by other routing mechanisms.

Despite these characteristics there is one major drawback of this approach: when limiting hop count (and when link pruning is disabled) we can only use links belonging to the shortest paths, so that we cannot fully benefit from existing longer and lightly loaded routes. Consider the situation shown in Figure 2.3, where we have two networks N and M interconnected using a short (two hop) path (A-E-D) and a longer path (A-B-C-D). If when establishing connections between network N and M we restrict ourselves only to the use of the shortest path (link pruning being disabled), the longer path remains unused. In such cases the RC approach is not beneficial compared even with static routing.

[7]The pruning process removes all infeasible links from the network graph representing a given topology.

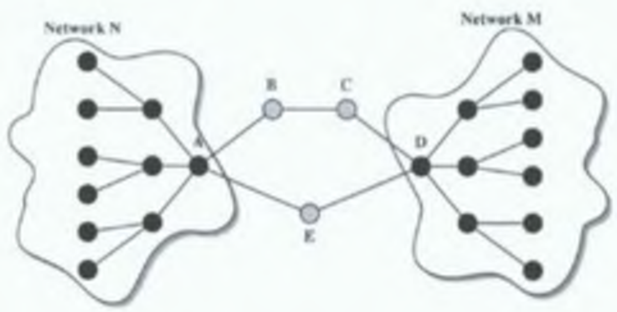


**Figure 2.3:** *An example when the shortest path approach performs poorly*

We can avoid scenarios as described above, simply by pruning links with insufficient resources first and calculating the min-hop path over a reduced topology. However it was observed [8, 133] that when state information is highly imprecise it is difficult to decide whether or not link has enough resources to accommodate the new request. When state information is imprecise, the pruning process can remove feasible as well as infeasible links, thereby degrading the overall performance [8, 133]. That is why we assume that pruning is disabled.

In this thesis we assume that inaccuracy of state information is present at all times, so pruning is disabled.

For some networks (such as that shown in Figure 2.3) the inefficient usage of resources by the RC approach can be overcome by admitting paths with hop count equal to the minimum plus one [98, 102]. However we are looking for general conclusions, and so such ad-hoc solutions are not considered here.

### 2.5.2 The load distributing approach (LD)

Considerable attention has also been given to LD algorithms [19, 65, 82]. Such load balancing is often achieved by the use of link cost metrics expressed as a convex function which increases with the link utilisation [19, 58, 65, 82, 100]. The benefit of using such a convex function for the link cost is intuitively clear: it favours paths on lightly loaded links over those on busy links, so that the load is balanced over all links

This approach to QoS routing has the following advantages

- It avoids congested links,
- It distributes load over the network, so that it can fully exploit longer paths,

It also has a number of drawbacks

- By using longer paths it consumes extra resources, and this can block future requests
- Due to the use of a main cost metric which depends on information about available resources, it requires up-to-date state information Otherwise considerable performance degradation is observed

QoS routing employing LD can be beneficial in situations such as shown in Figure 2 3, but in general it is vulnerable to inaccuracy of network state information, which can drastically degrade its performance, as observed in the ARPANET

## 2.6 Optimal Routing

The performance of QoS Routing is often compared with *optimal routing* (see [19]) So we need to understand what *optimal routing* is and why QoS routing methods applied in virtual circuit networks cannot perform as well as the optimal version

*Optimal routing* does not optimise the cost of any single path (as a typical QoS routing algorithm does) but rather optimises a global cost function such as the overall network throughput or the total delay It was shown in [144] that if there is only one class of users it makes little difference whether we minimise total delay or maximise total throughput

The optimisation of the global cost is extremely difficult to perform using on-line methods It requires global coordination of all network nodes and often rerouting of even an infinitesimal fraction of a flow over an alternative path

Such conditions are difficult to meet in real data networks, because often flow QoS constraints do not allow the splitting of data Also global synchronisation of network nodes is interrupted by *update policies* (see Section 3 4) which limit the communication overhead but introduce inaccuracy [10] Therefore, on-line routing algorithms can only mimic the behaviour of optimal routing methods as shown in [19]

*Optimal routing* can be treated as an instance of the *Multicommodity Flow* problem (MCF) [119] The multicommodity flow problem involves the simultaneous shipping of several different commodities (traffic classes) from their respective sources to their destinations in a single network so that the total amount of flow going through each edge is no more than its capacity (link bandwidth) Associated with each traffic class is a demand (requested bandwidth), which is the amount of that commodity that we wish to ship

## 2 6.1 Theoretical formulation of Optimal Routing

In this section we follow the formulation of optimal routing as presented in [29]

Let us assume that we operate on a weighted graph model $G = (V, E)$ (each node $n \in V$ and each link $e \in E$) Each link $e$ has assigned capacity $c(e)$ We denote by $N$ the number of nodes in the network and by $L$ the number of links We also denote by $W$ the set of all pairs $w = (s, d)$, where $s$ is a source and $d$ is a destination We denote by $P$ the set of all paths Each source–destination pair $w$ is interconnected using a set of paths $P_w$ We denote a flow $f$ as a function $f \quad P \rightarrow R^+$, and with each source–destination pair $w$ we associate a path flow vector $\overrightarrow{f} = \{f_p \,|\, p \in P_w, w \in W\}$ So the flow traversing edge $e$ can be defined as

$$f_e = \sum_{p \in P \ e \in p} f_p$$

With each pair $w$ we associate a rate $r_w$ which is the amount of flow traversing between this source–destination pair The flow $r_w$ can be arbitrary divided among paths $P_w$, so $\sum_{p \in P_w} f_p = r_w$ Finally for each link $e$ we specify a load–dependent cost function $cost_e(f)$ We assume that the cost function is nonnegative, differentiable and nondecreasing So the cost of path $p$ is given by

$$cost_p(f) = \sum_{e \in p} cost_e(f_e)$$

Now, we can define an optimal routing as a routing which minimises the total cost function This can be formulated as a non-linear program

$$\begin{aligned}
&Min \quad \sum_{e \in E} cost_e(f_e) && && (2\ 1)\\
&subject\ to && && (2\ 2)\\
&\sum_{p \in P_w} f_p = r_w && \forall w \in W && (2\ 3)\\
&f_e = \sum_{p \in P\ e \in p} f_p \leq c(e) && \forall e \in E && (2\ 4)\\
&f_p \geq 0 && \forall p \in P && (2\ 5)
\end{aligned}$$

## 2.6.2 Optimality condition

Let us define optimality conditions for one flow vector associated with a source–destination pair $w$ For that we need a definition of the first derivative length of the path Denote by $COST(f)$ the total cost function from (2 1)

$$COST(f) = \sum_{e \in E} cost_e(f_e) \tag{2 6}$$

So the first derivative $\frac{\partial COST(f)}{\partial f_p}$ of $COST(f)$ with respect to $f_p$ is represented by

$$\frac{\partial COST(f)}{\partial f_p} = \sum_{e \in p} cost'_e(f_e) \tag{2 7}$$

The first derivative $\frac{\partial COST(f)}{\partial f_p}$ can be considered as the length of the path $p$ when the length of each link is taken as the first derivative $cost'_e$ evaluated over the total flow $f_e$ Thus it can be called the first derivative length of path $p$

Should $\overrightarrow{f}^* = \{f_p^*\}$ be an optimal path flow vector for $w$, we cannot decrease the cost for this path flow So by shifting a small amount $\delta > 0$ of flow from path $p$ to $p'$ we can only increase the cost or at best keep it unchanged Using the first derivative path length we can write this change as

$$\delta\frac{\partial COST(\overrightarrow{f}^*)}{\partial f_{p'}} - \delta\frac{\partial COST(\overrightarrow{f}^*)}{\partial f_p}$$

Since this change cannot decrease the cost, we get

$$\frac{\partial COST(\overrightarrow{f}^*)}{\partial f_{p'}} \geq \frac{\partial COST(\overrightarrow{f}^*)}{\partial f_p} \tag{2 8}$$

So the path flow vector is optimal only when paths with minimum first derivative are used Since the local and global optima of convex function coincide (see [118]), the condition (2 8) is a necessary and sufficient to achieve global optimality of the objective function (2 1)

### 2.6.3 Optimal routing used in off-line optimisation

Optimisation of the global cost performed by optimal routing mechanisms is extremely difficult to perform using on-line methods That is why it is primarily used by traffic engineering tools as an off-line optimisation process In this section we present an example of such optimisation for two traffic classes, namely the QoS service classes and the Best Effort class This model is similar to the one presented in [50], but many other similar models can be found in the literature [70, 104, 139, 143] The original version of this model used flow definition as presented in [57] In this thesis we present a modified version of this model, assuming as in Section 2 6 1 that traffic flow can be split in an arbitrary way between the set of possible paths

This off-line optimisation considers only a single level of priority, however, the approach can be extended to multiple priority levels The QoS service classes get higher priority access to bandwidth than Best Effort traffic As a consequence, existence of the Best Effort traffic is transparent to the performance of the QoS service classes Thus this problem can be formulated as a multi criterion optimisation problem The first objective is the maximisation of bandwidth from QoS service classes, and the second objective is to maximise the Best Effort traffic subject to the constraint that the measure for the QoS traffic is optimal To make the problem more general, it is assumed that bandwidth is not optimised but rather the network revenue, which is a function of bandwidth [50]

Let us consider a weighted graph model $G = (V, E)$ as defined in Section 2 6 1 Traffic flows $f_w$ are classified into QoS and Best Effort We expect that there are several QoS service classes, with different requirements Also we assume that there is a single Best Effort service class The QoS group of service classes is denoted by $S_{QoS}$ With each pair $w$ we associate a rate $r_{w,s}$ which is the amount of flow traversing between this source–destination pair belonging to QoS class $s$

The objective to be maximised is *network revenue* It has two main components, $W_{QoS}$ and $W_{BE}$, for the QoS and Best Effort service class groups respectively

The QoS traffic produces revenue $W_{QoS}$ given by

$$W_{QoS} = \sum_{s \in S_{QoS}} \sum_{w \in W} \sum_{p \in P_w} e_{s,p} f_{s,w,p} \qquad (2\ 9)$$

where $f_{s,w,p}$ is the bandwidth carried on route $p$ between source–destination pair $w$ belonging to class $s$, and $e_{s,p}$ is the corresponding earnings per unit of carried bandwidth for QoS class $s$ over path $p$

The Best Effort traffic produces revenue given by

$$W_{BE} = \sum_{w \in W} e_{BE,\sigma} f_{BE,w} \tag{2 10}$$

where $f_{BE,w}$ is the total Best Effort flow or carried bandwidth for source–destination pair $w$, and $e_{BE,\sigma}$ is the corresponding earning rate We assume that the earning rate of Best Effort traffic does not depend on the chosen route

### 2 6 3 1 Routing for QoS Traffic

The problem of routing QoS flows can defined as a linear program [50]

$$Max \quad W_{QoS} \tag{2 11}$$

$$subject\ to \tag{2 12}$$

$$\sum_{p \in P_w} f_{s,w,p} \leq r_{w,s} \qquad \forall w \in W,\ \forall s \in S_{QoS}, \tag{2 13}$$

$$f_{e,QoS} = \sum_{s \in S_{QoS}} \sum_{w \in W} \sum_{p \in P_w\ e \in p} f_{s,w,p} \leq c(e) \qquad \forall e \in E, \tag{2 14}$$

$$f_{s,w,p} \geq 0 \qquad \forall p \in P,\ \forall s \in S_{QoS},\ \forall w \in W \tag{2 15}$$

The $f_{e\ QoS}$ is the amount of bandwidth assigned to QoS flows on link $e$ Because the QoS flows are given priority over best effort flows, this reserved bandwidth will not be available for best effort traffic Here the $\sum_{p \in P_w} f_{s\ w\ p}$ is the flow traversing between source–destination pair $w$ and the slack in Eq (2 13) represents the loss, which may be ascribed to admission control

### 2 6 3 2 Routing for Best Effort Traffic

The problem of routing best effort traffic can defined as a linear program

$$Max \quad W_{BE} \tag{2 16}$$

$$subject\ to \tag{2 17}$$

$$\sum_{p \in P_w} f_{BE,w,p} \leq r_{w\ BE} \qquad \forall w \in W,\ \forall s \in S_{QoS}, \tag{2 18}$$

$$\sum_{w \in W} \sum_{p \in P_w\ e \in p} f_{BE,w,p} \leq c(e) - f_{e,QoS} \qquad \forall e \in E, \tag{2 19}$$

$$f_{BE,w\ p} \geq 0 \qquad \forall p \in P,\ \forall w \in W \tag{2 20}$$

Due to QoS flows being given priority, best effort flows can utilise only the residual bandwidth following the phase of QoS flow optimisation (see Eq (2 19))

### 2 6 3 3 Combined routing for QoS and Best Effort Traffic

The problem of reserving bandwidth for QoS and Best Effort traffic can be solved separately, by reserving bandwidth first for QoS flows and later using the remaining bandwidth to perform reservation for Best Effort flows However this method does not permit QoS flows to be reassigned without decreasing the revenue to maximise the bandwidth of best effort flows To remove this flaw we define a problem of combined reservation

Let us assume that $W^*_{QoS} = \max\ W_{QoS}$ is the optimum revenue generated by assigning QoS flows Thus the combined problem can be defined as [50]

$$Max \quad W_{BE} \tag{2 21}$$

$$subject\ to \tag{2 22}$$

$$W^*_{QoS} = \max\ W_{QoS} \tag{2 23}$$

$$\sum_{p \in P_w} f_{BE,w,p} \leq r_{w,BE} \quad \forall w \in W,\ \forall s \in S_{QoS}, \tag{2 24}$$

$$\sum_{w \in W} \sum_{p \in P_w\ e \in p} [f_{BE\ w,p} + \sum_{s \in S_{QoS}} f_{s,w,p}] \leq c(e) \quad \forall e \in E, \tag{2 25}$$

$$f_{BE,w,p} \geq 0 \quad \forall p \in P,\ \forall w \in W \tag{2 26}$$

In Eq (2 23) $W^*_{QoS}$ is obtained by solving the problem for the QoS traffic in isolation In the second step the values of QoS flows can change to improve the revenue from best effort traffic while preserving optimal revenue of QoS flows

In general to get optimally selected routes requires the solution of linear programs [1, 37] using for such tools as CPLEX[8] or PPRN [37] Or we can use approximate methods, which improve the algorithm running time and produce results very close to the optimal solution [20, 55, 63, 71]

### 2.6.4 On-line suboptimal routing

Optimal routing requires high computations and also makes assumptions which cannot be satisfied in real networks For example as stated in [29]

> *Implicit in flow models is the assumption that the statistics of the traffic entering the network do not change over time*

However traffic patterns often vary, and therefore the short-term performance of such methods may be suboptimal [138] In the following we present various approaches to on-line routing which imitate the theoretically optimal methods

[8]CPLEX Optimization Inc , *CPLEX User s Manual, Version 3 0,* Incline Village, NV, (1994)

#### 2 6 4 1 Distributed Minimum Delay Routing

Robert Gallager in [60] proposed a distributed routing algorithm which minimises the total delay for input traffic with stationary characteristics This model also assumes that the incoming traffic at the node can be split in an arbitrary way between outgoing links The algorithm is applied at each node and is based on information about the first derivative delay to each destination

It was shown in [29] that when traffic is optimally distributed, the cost of each path carrying a flow between the source–destination pair $w$ must have an equal first derivative cost path (see Eq 2 8) This fact is exploited in the distributed algorithm proposed by Gallager In this algorithm each node shifts a small fraction of the flow from links with high first derivative cost to links with small marginal (first derivative) cost The algorithm converges to an optimal solution [60]

#### 2 6 4 2 Selfish routing

Typically during route selection each node simply chooses the paths of minimum cost rather than paths which minimise the total cost This strategy is based on individual rationality rather than group rationality [46] Each user attempts to maximise its use of the resource with no regard to the effects of its action on the other users So the paths with minimal cost are selected rather than the minimal first derivative cost This usually works well when there is plenty of available bandwidth However, when the provided bandwidth is less than required, the self-interested actions of individuals may lead to congestion

It was shown in [126] that if optimal routing minimises the total latency, the total latency produced by selfish routing decisions can be arbitrary larger than the minimum possible total latency, although it is not more than the total latency incurred by optimal routing where the load is doubled [126] This upper bound does not look optimistic, however the Internet is not centrally administered by any authority, so network users are free to act according to their own interests

Thus results in a "selfishly motivated" assignment of the traffic, which does not optimise the total cost function

### 2 6 4 3 Throughput competitive routing

The two routing methods described above assume that flows can be split in an arbitrary way, which is rarely allowed for flows requiring QoS guarantees in real networks The more realistic scenario of a routing mechanism trying to mimic the behaviour of optimal routing has been presented in [19] This model deals with unsplittable flows and thanks to appropriately selected connection admission control mechanisms it tries to achieve a throughput as close as possible to the optimal solution However this model assumes that the duration of each connection is known *a priori* (at the moment when request arrives at the node)

In [19] the authors use the concept of "competitive ratio" (originally introduced in [129]) to compare the performance of their algorithm with the optimal solution The *competitive throughput ratio* is the supremum, over all possible input sequences, of the throughput achieved by the optimal off-line algorithm to the throughput achieved by the on-line algorithm under consideration [119] The optimal off-line algorithm is not equivalent to *optimal routing* since flows are unsplittable, it is rather an algorithm which has *a priori* knowledge of the entire request sequence, and the considered on-line algorithm deals with each request one by one and does not possess any knowledge about future requests

The algorithm presented in [19] achieves a competitive ratio of $O(\log NT)$, where $T$ is the maximum duration of the connection and $N$ is the number of nodes in the network The proposed solution uses an approach used to solve multicommodity flow problem by repeated rerouting of flows onto shortest paths [86, 93, 130] By defining the cost function to be exponential in the current link usage, these algorithms solve the approximate multicommodity flow problem in polynomial time However this solutions use repeated rerouting, so they can be implemented only as off-line methods

Let us consider a weighted graph model $G = (V, E)$ as defined in Section 2 6 1 The considered algorithm deals with each request one by one, so the $i$th request $\beta_i$ can be defined as

$$\beta_i = \left(w_i, r_i(\tau), T_i^s, T_i^f, \rho_i\right),$$

where $w_i$ is the source–destination pair of the $i$th request, $r_i(\tau)$ is the traffic rate at time $\tau$ requested by the connection and $\rho_i$ is the profit acquired for accepting this request $T_i^s$ and $T_i^f$ are the connection starting and finish time, so we can assume that $r_i(\tau) = 0$ for $\tau < T_i^s$ and $\tau > T_i^f$ Assuming that if the request is accepted the connection is assigned one path $p_i$ (where the flow cannot be split), the *relative utilisation* on edge $e$ before considering $k$th request and after considering request $k - 1$ can be defined as

$$u_e(\tau, k) = \sum_{\substack{e \in p_i \\ i<k}} \frac{r_i(\tau)}{c(e)}$$

In the algorithm, every link $e$ has associated with it a cost function that is exponential to the bandwidth utilisation

$$cost_e(\tau, k) = c(e)(\mu^{u_e(\tau,k)} - 1)$$

A new connection is admitted into the network only if there exists a path whose accumulated cost over the duration of the connection does not exceed the profit that is measured by the bandwidth-duration product of the connection (see Table 2 1)

**Table 2 1** *Throughput-competitive routing algorithm*

For each request $\beta_i = (w_i, r_i(\tau), T_i^s, T_i^f, \rho_i)$ check
if $\exists$ path $p$ in $P_w$ such

$$\sum_{e \in p} \sum_{T_i^s \leq \tau \leq T_i^f} \frac{r(\tau)}{c(e)} cost_e(\tau, i) \leq \rho_i$$

**then** route the connection on $p$, and set

$$u_e(\tau, i+1) = u_e(\tau, i) + \frac{r_i(\tau)}{c(e)} \qquad \forall e \in p, \; T_i^s \leq \tau \leq T_i^f,$$

**else** block the connection

The algorithm achieves a competitive ratio of $O(\log NT)$, as shown in [19]

### 2 6 4 4 Is suboptimal routing practical?

The above approaches to mimicing the behaviour of optimal routing possess one or more of the following shortcomings

- repeated rerouting of flows is required,
- flow splitting is required,
- *a priori* knowledge about connection duration is required,
- precise information about links cost is required,
- only very small flows are accepted

For example, if there is no *a priori* knowledge about connection holding times, there is no throughput competitive strategy (because an algorithm without such knowledge can reject connections with infinite duration, which would be accepted by the optimal off-line algorithm [119])

The routing algorithms performing routing decisions on-line in existing networks usually deal with unsplittable flows with unknown duration and are provided only with inaccurate information Therefore algorithms trying to mimic the behaviour of optimal methods employed in real networks often perform worse than algorithms computing just paths with the minimum number of hops So such algorithms cannot be employed in existing data networks

## 2.7 Related topics

In this section we present techniques related to QoS routing Some of these such as Constraint Based Routing or Traffic Engineering use similar kinds of mechanisms to QoS routing but also extend them to address specific technological problems Other techniques often accompany the QoS routing or provide new architectures with QoS routing support

### 2.7.1 Traffic Engineering and Constraint Based Routing

The Constraint Based Routing [79] is a mechanism that supports Traffic Engineering [15] Thus these terms are often used interchangeably These techniques are considered as more general than QoS routing since apart from QoS constraints they may consider also policy and management constraints Also they may engineer the network traffic without considering QoS requirements

In general Traffic Engineering can be characterised as "the process of arranging how traffic flows through the network so that congestion caused by uneven network utilisation can be avoided" [155] This definition, however, does not capture the whole set of mechanisms involved in engineering the network traffic So informally we define Traffic Engineering as the set of all processes to control the network traffic either automatic or performed manually Informally we can also define Constraint Based Routing as the process of automating the task of Traffic Engineering

Different methodologies have been proposed for Traffic Engineering A complete taxonomy of this methods can be found in [15] In this thesis we classify them into three basic types [15, 51]

- Time-dependent - these traffic control algorithms optimise network resource utilisation in response to long-time-scale traffic variations [2] They make no attempt to adapt to random short-term traffic variations or changing network conditions An example of such a time-dependent algorithm is the optimisation-based centralised control algorithm proposed by Mitra [50] We describe the method from [50] in Section 2 6 3 as it performs off-line optimisation of the global cost function So time-dependent algorithms are used off-line and utilise complicated mathematical tools to engineer the traffic based on historical information collected on-line

- State-dependent - these traffic control algorithms adapt network traffic to relatively fast network state changes Examples of such a state-dependent approach can be found in [13, 51, 87, 140] The adaptation to changes in network state can be based on information flooded by the routers, or collected by probe packets, or gathered by the management system Constraint-based routing and QoS routing can be considered as examples of the state-dependent class

- Event-dependent - these algorithms search for new paths based only on the occurrence of events such as when a call setup encounters congested or blocked links So they use learning techniques to optimise network resource utilisation This class of methods is different from the state-dependent class because the dissemination of global state information is not required Examples of such event-dependent methods of Traffic Engineering can be found in [106, 108]

Despite the variety of methods that have been proposed for Traffic Engineering most enterprises were skeptical about its merits, because the offered technologies were complicated and and were considered too risky [31] So the solutions existing on the market for the past few years are rather simple, such as

- solutions balancing for load among multiple Web and application servers, based on
  - Load balancing devices and Layer 4 - 7 switches, such as those from Alteon (now part of Nortel[9]), ArrowPoint (now part of Cisco[10]), F5[11] and Foundry[12],
  - optimising services from vendors like Akamai[13],
- solutions for utilising multihoming (Multihoming is the technique of connecting a network to the Internet via two or more ISPs ), such as
  - BGP-based route control offered by netVMG[14], InterNAP[15], Opnix [16], Proficient Networks [17], RouteScience [18], SockEye Networks[19],
  - link-based controllers such as *BigIP Link Controller* produced by F5 or Radwares *LinkProof*[20]

Recently, however, there are new emerging solutions for MPLS Traffic Engineering such as MATE offered by Cariden[21], solutions by Alcatel[22], PARC Technologies[23], and Foundry, and platforms enabling such solutions provided by Cisco, Juniper Networks[24], and Avici Systems[25] Such MPLS-enabled mechanisms seem to provide efficient control over the traffic and will probably dominate future traffic engineering initiatives

[9]Nortel, http //www nortelnetworks com/
[10]Cisco, http //www cisco com/
[11]F5, http //www f5 com/
[12]Foundry, http //www foundrynet com/
[13]Akamai, http //www akamai com/
[14]netVMG, http //www netvmg com/
[15]InterNAP, http //www internap com/
[16]Opnix, http //www opnix com/
[17]Proficient Networks, http //www proficientnetworks com/
[18]RouteScience, http //www routescience com/
[19]SockEye Networks, http //www sockeye com/
[20]Radware http //www radware com/
[21]Cariden, http //www cariden com/
[22]Alcatel, http //www alcatel com/
[23]PARC Technologies, http //www parc-technologies com/
[24]Juniper Networks, http //www juniper net/
[25]Avici Systems, http //www avici com/

As we can see in the above examples, Traffic Engineering includes a variety of different mechanisms for static or dynamic traffic control. QoS routing can be considered as a subclass of traffic engineering methods, which performs short term traffic optimisation with respect to QoS constraints.

### 2.7.2 Admission Control

QoS routing should efficiently utilise network resources. However, often an appropriate path selection algorithm is not enough. Sometimes if a flow needs too much resources, we may decide to reject it even if the network has the capability to accept it. Such a decision may be taken because by rejecting the flow we can achieve a higher network throughput and satisfy further requests of other users. So, QoS routing is often accompanied by higher level admission control mechanisms to perform efficient management of networking resources [18, 19, 61, 65, 82, 119, 137].

### 2.7.3 Resource Reservation

QoS routing and resource reservation are closely connected techniques. To provide QoS guarantees to user flows, we need to compute a feasible path from source to destination and reserve the resources along the path. The first task is done by QoS routing, while the second one is done by resource reservation protocols such as RSVP [33].

Resource Reservation Protocol (RSVP) [33] is a “soft state” signaling protocol. It supports receiver-initiated establishment of resource reservations for both multicast and unicast flows. RSVP was originally developed as a signaling protocol within the *Integrated Services* framework for applications to communicate QoS requirements to the network and for the network to reserve relevant resources to satisfy the QoS requirements [33].

The reservation process uses two fundamental RSVP message types: RESV and PATH The PATH message is send by the sender to the receiver specifying the characteristics of the traffic Every intermediate router along the path forwards the PATH message to the next hop determined by the routing protocol Upon receiving a PATH message, the receiver responds with a RESV message to request resources for the flow Every intermediate router along the path can reject or accept the request of the RESV message If the request is rejected, the router will send an error message to the receiver, and the signaling process will terminate If the request is accepted, link bandwidth and buffer space are allocated for the flow and the related flow state information will be installed in the router

The "soft state" approach used in RSVP means that the reservation is created and periodically refreshed by PATH and RESV messages The state is deleted if no matching refresh messages arrive before the expiration of update interval

In conclusion RSVP is a receiver-oriented mechanism for resource reservations for both unicast and multicast applications However it works in "soft state" fashion, which is considered to be a bottleneck in backbone networks [155]

Recently, RSVP has been modified and extended in several ways to mitigate the scaling problems As a result, it is becoming a versatile signaling protocol for the Internet For example, RSVP has been extended to reserve resources for aggregation of flows, to set up MPLS explicit label switched paths [14], and to perform other signaling functions within the Internet There are also a number of proposals to reduce the amount of refresh messages required to maintain established RSVP sessions [26]

## 2 7.4 Integrated Services and Differentiated Services

The Internet Engineering Task Force (IETF)[26] in recent years developed two service models for providing QoS support in the Internet Integrated Services [32] and Differentiated Services [30]

[26]The Internet Engineering Task Force, `http //www ietf org/`

### 2 7 4 1 Integrated Services (IntServ)

The Integrated Services (IntServ) [32] model requires resources, such as bandwidth and buffers, to be reserved a priori for a given traffic flow to ensure that the quality of service requested by the traffic flow is satisfied The integrated services model proposes two service classes in addition to best effort service, they are

- **guaranteed service** [135] – this is used for applications requiring bounded packet delivery time This is accomplished by controlling the queuing delay on network elements along the data flow path
- **controlled load service** [154] – this is used for applications that can tolerate some delay and are sensitive to traffic overload conditions Controlled-load service has been designed to provide approximately the same service as best-effort service in a lightly loaded network regardless of actual network conditions

The philosophy of the IntServ model is that "there is an inescapable requirement for routers to be able to reserve resources in order to provide special QoS for specific user packet streams, or flows This in turn requires flow-specific state in the routers" [32]

Int-Serv is implemented by four components

1 signaling protocol (e g RSVP - described next),

2 admission control routine,

3 classifier,

4 packet scheduler

Applications requiring guaranteed service or controlled-load service must set up the paths and reserve resources before transmitting their data The admission control routines will decide whether a request for resources can be granted When a router receives a packet, the classifier will perform a multi-field (MF) classification and put the packet in a specific queue based on the classification result The packet scheduler will then schedule the packet appropriately to meet its QoS requirements

The IntServ architecture is quite computationally demanding and the most commonly mentioned problems of IntServ are

- The amount of state information increases proportionally with the number of flows This places a huge storage and processing overhead on the backbone routers Therefore, this architecture does not scale well in the Internet core

- The computational requirement on routers is high All routers must support RSVP, admission control, MF classification and packet scheduling

### 2 7 4 2 Differentiated Services (DiffServ)

The initial goal of the Differentiated Services (DiffServ) [30] effort within the IETF was to develop scalable mechanisms for classification of traffic flows into flow aggregates, which allows each flow aggregate to be treated differently [30] Thus DiffServ provides an alternative mechanism for service differentiation in the Internet that alleviates the scalability issues encountered with the IntServ model

The IETF Diff-Serv working group has defined a Differentiated Services field in the IP header (DS field) [110] The DS field is the part of the IP header formerly known as the Type of Service (ToS) octet The DS field is used to indicate the forwarding treatment that a packet should receive at a node The DiffServ working group has also specified a number of Per-Hop Behaviour (PHB) groups [111] Using the PHBs, several classes of services can be defined using different classification, policing, shaping and scheduling rules

For an end user of network services to receive Differentiated Services from its Internet Service Provider (ISP), it may be necessary to have a Service Level Agreement (SLA) with the ISP The SLA specifies a Traffic Conditioning Agreement (TCA) which defines classifier rules as well as metering, marking, discarding, and shaping rules

Packets are classified, and possibly policed and shaped at the ingress to a DiffServ network When a packet traverses the boundary between different DiffServ domains, the DS field of the packet may be re-marked according to existing agreements between the domains

Using the PHBs and different classification, policing, shaping, several classes of services can be defined such as

- **Assured service** [76] – employed for applications requiring better reliability than best-effort service

- **Premium service** [112] – used for applications requiring low-delay and low-jitter service

Differentiated Services allows only a finite number of service classes to be indicated by the DS field The main advantage of the DiffServ approach relative to the IntServ model is scalability Resources are allocated on a per-class basis and the amount of state information is proportional to the number of classes rather than to the number of flows

## 2 7 5 Integrated Services over DiffServ Networks

In order to exploit the advantages of per flow QoS guarantees of the IntServ model and good scalability in the backbone of the DiffServ model, the interconnection of IntServ and DiffServ was proposed [22, 24, 27, 34] In this model QoS is provided by applying the end-to-end IntServ model across a network containing one or more DiffServ regions The DiffServ regions may, but are not required to, participate in end-to-end RSVP signaling for the purpose of optimising resource allocation and supporting admission control

The IntServ over DiffServ model seems to be the most promising QoS model to be used in the future Internet, since it provides per-flow reservations while maintaining scalability to large networks

## 2 7 6 MPLS

Multi-Protocol Label Switching (MPLS) [125] is one of several initiatives to enable packet delivery for a future converged networks, by combining the attributes of Layer 2 switching and Layer 3 routing into a single entity

The basic idea of MPLS is to forward the packets based on a short, fixed length identifier termed a "label", instead of the network layer address The labels are assigned to the packets at the ingress node of an MPLS domain Inside the MPLS domain, the labels attached to packets are used to make forwarding decisions Thus, MPLS uses indexing instead of a longest prefix match as in conventional IP routing The labels are finally popped out from the packets when they leave the MPLS domain at the egress nodes By doing this, the efficiency of packet forwarding is greatly improved, since longest prefix match process (whose complicated computations slow down the IP routing [74, 127, 147]) is avoided

The motivation for MPLS was to facilitate fast packet switching and forwarding, but currently the main incentive for utilising MPLS is to support Traffic Engineering and provide Quality of Service [13, 16, 17, 51, 87, 140, 145] (see also Section 2 7 1) The MPLS technology supports explicit routing, which can be used to optimise the utilisation, also multiple paths can be used simultaneously to improve performance from a given source to a destination The labels used in MPLS also allow different forwarding rules (and hence different quality of service) to be assigned to different flows

Thus MPLS is a promising technology for managing flows in networks using such mechanisms as QoS routing, Constraint Based Routing and Traffic Engineering

### 2.7.7 Active Networks

Active Networks [121, 141, 142] is a relatively new concept, where a network is not just a passive carrier of data but also a more general model capable of performing customised computations on carried data. Since it is able to provide various operations on network flows, such technology can be used as a platform for gradual employment of QoS aware routing algorithms [152].

Active Networks are "active" in two ways: routers and switches within the network can perform computations on user data flowing through them, and users can program the network, by supplying their own programs to perform customised computations [141]. Traditional data networks provide a transport mechanism to transfer data from one end system to another. In contrast to that active networks allow the network nodes to perform computations on the data and also allow their users to inject customised programs into the nodes of the network, that may modify, store or redirect the user data flowing through the network. These programmable networks open many new possibilities for applications that could not be implemented in traditional data networks. For example, there may be a multicast video stream in which the compression rate can be modified when the network bandwidth available is less than required by the original stream compression (for example at the boundary between wireless and copper networks).

Opinion in the research community is divided as to whether or not Active Networks are useful [121]. The argument against active networks is that the key of Internet success was its simplicity and by making the networks active we can destroy this idea. As George Gilder said [69] "In a world of dumb terminals and telephones, networks had to be smart. But in a world of smart terminals, networks have to be dumb." The argument supporting active networks is that it is a very promising and innovative idea and a variety of useful network services that require processing of data will be possible, which can lead to better end-to-end performance being perceived by users [92]

## 2.8 Summary

Most current routing schemes select paths considering only a single metric such as hopcount or cost The aim of such routing algorithms is simply to provide connectivity between network nodes QoS routing selects a path for each connection to satisfy diverse performance requirements and to optimise resource usage Thus it enables the deployment of new services the require QoS support, while optimising the overall network performance However, QoS routing has to deal with some challenging issues that are not present in traditional routing, including the scalable dissemination of dynamic (state-dependent) information, state aggregation and the computation of constrained paths The search for effective QoS routing methods has been ongoing since the early days of the Internet

In this chapter, we have presented the components of a QoS routing architecture We have also summarised the historical development of such methods and have described the theoretically optimal solutions and technologies supporting the deployment of QoS routing

We have also included an overview of the challenges faced by QoS routing These problems will be addressed in the rest of the thesis In order to allow gradual rollout of QoS routing methods, they are likely to be introduced by extending existing link state routing protocols such as OSPF Therefore we restrict our attention to the link state QoS routing architecture which we believe to provide the most practical way for the implementation and deployment of QoS routing features in the Internet

# CHAPTER 3

# Model for QoS routing

In this chapter we present the routing model used in our experiments The details of the model can differ in each experiment but the principles of this model are preserved We focus on a family of link state QoS routing mechanisms These schemes are able to find paths satisfying QoS constraints imposed by the users, efficiently utilise network resources, and accommodate to periods of transient overload, link failure, or general congestion However, QoS routing protocols can impose a significant bandwidth and processing load on the network, since each router must maintain its own view of the available link resources, distribute link state information to other routers, and compute and establish routes for new flow requests

In general the link state routing mechanisms can be summarised as follows For each link the bandwidth and optionally the cost value are advertised using a flooding mechanism or a spanning tree The frequency of such advertisements is controlled by the link state update policy Upon receiving a route request the routing algorithm computes a path If the reservation along a chosen path can be realised, the request is accepted, otherwise it is rejected Once the bandwidth resources are reserved on each link in the route, the network admits the flow, committing requested bandwidth on each link in the path for the duration of the flow

In the following part of this chapter we provide detailed information about all elements of this model

## 3.1 Routing algorithms

Since predictable communication performance relies on having some sort of throughput guarantee, our routing model views bandwidth as the primary traffic metric for defining both application QoS and network resources Although application requirements and network load may be characterised by several other dynamic parameters, including delay and loss, initial deployments of QoS routing are likely to focus simply on bandwidth to reduce algorithmic complexity

In this thesis we focus primarily on two approaches to QoS routing, namely the Load Distributing (LD) approach and the Resource Conserving (RC) approach (introduced in Section 2 5) These are represented by two basic QoS routing algorithms which consider bandwidth requirements

**widest-shortest path** (WSP) [9] - selects the shortest (min-hop) path, but if there are several such paths, the one with the largest residual bandwidth is chosen This algorithm is a typical example of the RC approach

**exponential cost function routing** (EXP) - selects the least cost path, where the link cost is expressed as an exponential function of its utilisation The choice of an exponential link cost function allows paths to avoid congested links, and is a typical example of the LD approach

These algorithms can also be supported by a pruning mechanism By pruning any infeasible links (subject to stale information), the source performs a preliminary form of resource reservation to avoid selecting a route that cannot support the new flow The prunıng process produces a sparser network graph consıstıng entırely of feasıble lınks However, the process of prunıng lınks ın the presence of stale lınk state ınformatıon may result ın routıng errors, sınce the source may ıncorrectly prune a lınk that could actually support the new flow or use a lınk that does not support the new flow Thus prunıng ıs not recommended when state ınformatıon ıs ımprecıse [8, 133] and ın thıs thesıs we assume that ıt ıs dısabled

### 3.1.1 Network model

We operate on a weıghted graph model $G = (V, E)$ (each node $n \in V$ and each lınk $e \in E$) Each lınk $e$ has assıgned capacıty $c(e)$ and an amount $r(e)$ of thıs bandwıdth ıs already reserved and cannot be assıgned to new connectıons So the current lınk utılısatıon $u(e)$ can be expressed as $r(e)/c(e)$ The route computıng algorıthm deals wıth each request ın turn, so the $i$th request can be defıned as $(s_i, d_i, r_i)$, where $s_i$ ıs the source, $d_i$ ıs the destınatıon, and $r_i$ ıs the traffıc rate specıfıed ın the request The traffıc rate $r_i$ represents eıther a peak, average, or effectıve bandwıdth, dependıng on the applıed admıssıon control polıcy

### 3 1.2 Lınk cost

For each lınk $e$ we specıfy a load-dependent cost functıon $cost(e)$ So the cost of path $p$ ıs gıven by

$$cost(e) = \sum_{e \in p} cost(e)$$

We also assume that the values of lınk cost metrıc values belong to the ınterval $< MIN, MAX >$ In our experıment we use a range between 1 and 65535, as ıs used ın OSPF [105] So for the WSP and the EXP algorıthms the correspondıng lınk cost functıons are defıned as follows

### 3 1 2 1 Link cost of WSP algorithm

The WSP algorithm has a strong preference for minimum-hop routes So its primary cost is constant, and we assume that

$$\text{cost}_{\text{WSP}}(e) = \text{C} \tag{3 1}$$

where C is a constant (see Figure 3 1) This constant can also be configured administratively or it can be proportional to the inverse of link capacity (as suggested by Cisco[1])

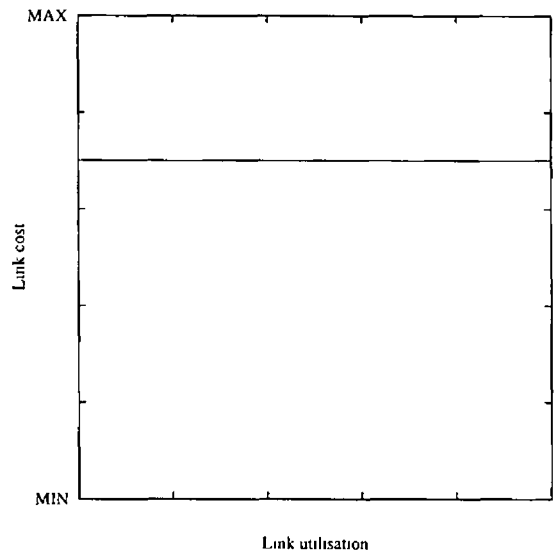


**Figure 3 1** *Link cost function of the WSP algorithm*

Such a cost function allows routers to select set of min-hop paths From this set the path with largest residual bandwidth is chosen, or in other words the path $p$ with maximum $\min_{e \in p}\{c(e) - r(e)\}$

### 3 1 2 2 Link cost of EXP algorithm

The EXP algorithm uses link cost expressed as an exponential function of its utilisation The choice of an exponential link cost function allows paths to avoid congested links In [19] it was theoretically shown that the network load can be

[1]http //www cisco com/warp/public/104/2 html

dıstrıbuted ın an almost optımal way by the use of an exponentıal lınk cost functıon We use a normalısed exponentıal cost functıon of the form

$$\exp(e) = \frac{A^{u(e)} - 1}{A - 1} \tag{3 2}$$

where $A$ ıs a large posıtıve constant (see [19, 65])

To allow the EXP lınk cost functıon to use the whole range of lınk cost values we assume that ıts fınal formula ıs gıven by

$$\text{cost}_{\text{EXP}}(e) = MIN + (MAX - MIN)\exp(e) \tag{3 3}$$

An example of thıs cost functıon ıs shown ın Fıgure 3 2

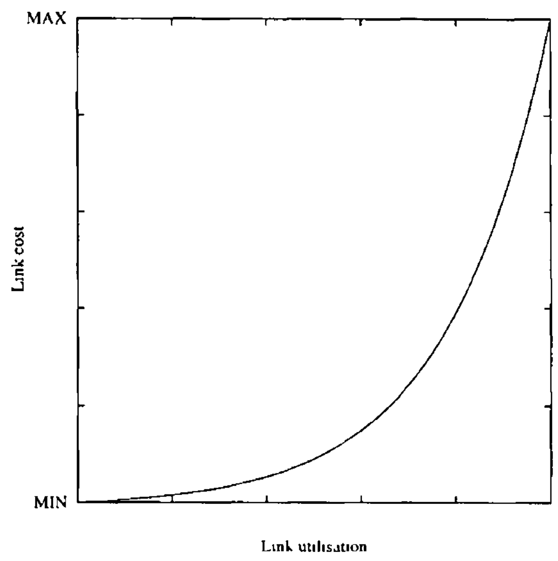


**Fıgure 3 2** *Lınk cost functıon of the EXP algorıthm*

## 3.2 Network Topology

Researchers have evaluated the performance of dıfferent routıng approaches on sımulated networks wıth varıous topologıes such as ISP [10, 160], MCI [98], random topologıes [120, 133], regular topologıes [133], and others [66, 102] However the topology of the Internet ıs dıffıcult to characterıse because ıt ıs constantly

**Table 3.1:** *Average node degree of network topologies used in our experiments*

| | Average node degree |
|---|---|
| ISP topology | 3.33 |
| Tree topology | 1.9 (extended to 7.6) |
| Low connectivity topology | 2.3 |
| High connectivity topology | 4.96 |
| Two-level hierarchical topology | 3 |

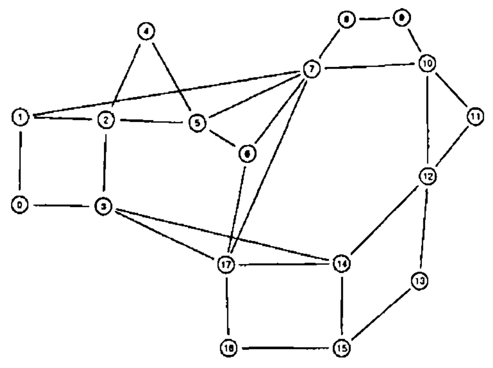
changing [56], so selecting an algorithm on the basis of its performance on a limited set of topologies should not be recommended.

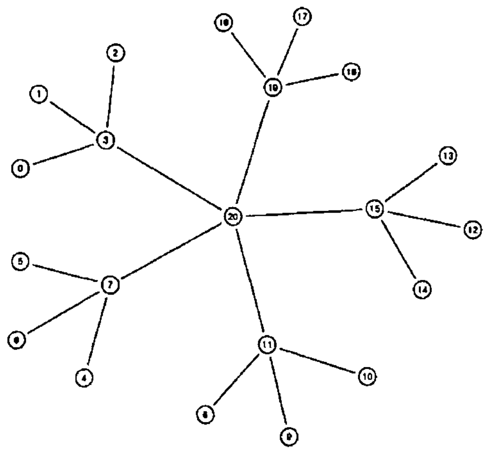
In this thesis we use the following topologies:

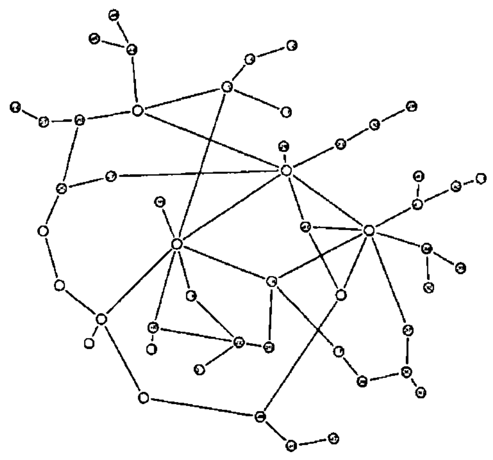
- the ISP topology [10, 160] – Figure 3.3(a);
- the tree topology with a minimal number of links, which we extend by adding links, to model various connectivity levels – Figure 3.3(b);
- two randomly generated topologies each with a different level of connectivity created using the GT-ITM software [161]– Figures 3.3(c) and 3.3(d);
- a randomly generated network with two hierarchical levels created using the GT-ITM software [161]– Figure 3.3(e);

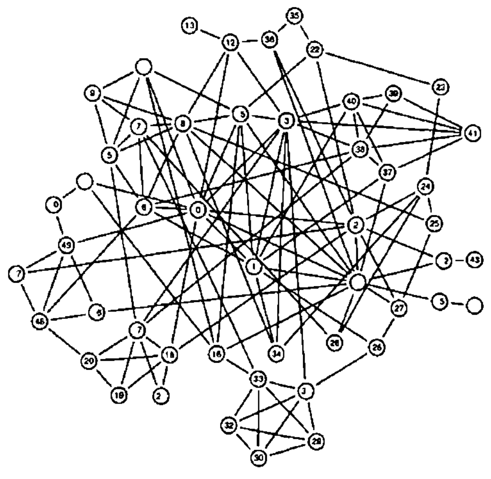
We denote by $N$ the number of nodes and by $L$ the number of links in the network. In all networks links are assumed to be bidirectional. Thus the average node degree is equal to $2L/N$. The corresponding values of average node degree for each network topology are shown in Table 3.1.

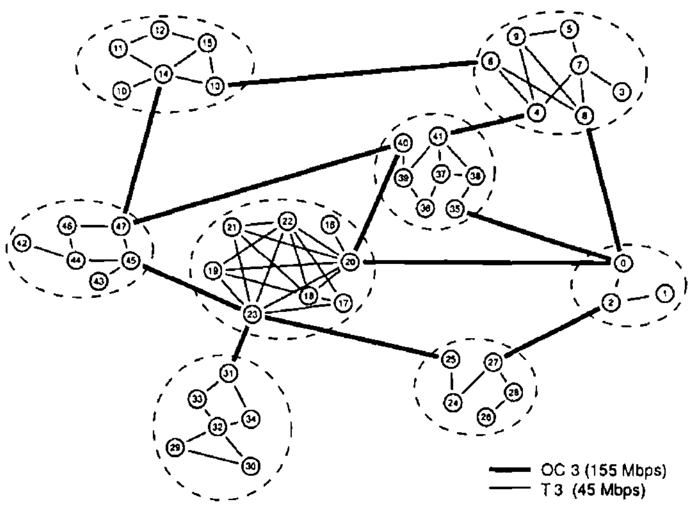
We use the tree topology to compare routing performance over networks with various connectivity levels. We start with a base network of average node degree 1.9. By randomly adding links to the base network we progressively effect a change of average node degree from 1.9 to 7.6. Other researchers usually utilise topologies with a degree of around four, and rarely report results for degrees higher than six or less than three (e.g., [120, 133]). Thus we study a wider range of connectivities.


(a) ISP


(b) Tree topology


(c) Low connectivity topology


(d) High connectivity topology



(e) Two-level hierarchical topology

**Figure 3 3** *Topologies used in simulations*

We do not consider regular topologies such as the cube [133], because they are infrequently used in real networks They feature many min-hop paths between source and destination, which has a positive impact when algorithms are restricted to a set of min-hop paths. However, such regularities are usually not present in real networks.

## 3.3 Traffic model

In our experiments we model two types of traffic, uniform and bursty traffic. For uniform traffic the requests arrive at each node independently according to a Poisson distribution with rate $\lambda$ and have exponentially distributed holding times with mean value $1/\mu$. To model bursty traffic we use interarrival times following a Weibull distribution [54] with shape parameter $c = 0.7$ and connection durations following a Pareto distribution [117] with shape parameter $a = 2.5$.

For both traffic patterns the requested amount of bandwidth is uniformly distributed over the interval: $[64kb/s, 6Mb/s]$, with mean value $B = 3.32Mb/s$.

Thus the load offered to the network is [133] is given by:

$$\rho = \frac{\lambda N^a B h'}{\mu L C} \tag{3.4}$$

where:

| | |
|---|---|
| $N$ | is the number of nodes in the network, |
| $N^a$ | is the number of nodes generating traffic, |
| $L$ | is the number of links in the network, |
| $C$ | is the links capacity, |
| $B$ | is the mean requested amount of bandwidth, |
| $h'$ | is the average hop count. |

The average shortest hop count ($h'$) is calculated over all source-destination pairs. If we assume that $\pi_{sd}$ stands for the minimum number of hops separating source $s$ with destination $d$, the average shortest hop count can be calculated as follows

$$h' = \frac{\sum_{s=1}^{N} \sum_{d=1,\ d\neq s}^{N} \pi_{sd}}{N(N-1)} \qquad (3\ 5)$$

For example $h' = 3\ 04762$ for the base network shown in Figure 3 3(b)

In our experiments we adjust $\lambda$ to produce the required offered load and the mean connection holding time is fixed

## 3.4 Update policies

Despite the apparent complexity of QoS routing, the path-selection and admission control framework also offers network designers a considerable amount of latitude in limiting overheads In particular, the network can control the complexity of the routing algorithm itself, as well as the frequency of route computation and link state update messages Information can be propagated in a periodic fashion or in response to a significant change in the link state The set of rules governing the frequency of link state advertisements is called the *update policy* There are several possible link state update policies [10, 133, 160], and we have classified them as follows

**timer based policy** – uses a timer to control the frequency of link state advertisements So the advertisements are triggered at the end of every refresh period This approach allows the frequency of updates to be controlled very precisely but long update intervals are likely to produce fluctuations in link utilisation [133] (as described in detail in Section 2 2 4)

**utilisation change based policies** – are used to send link state updates only if the link utilisation changes Nevertheless the link load can change very frequently To limit the flooding frequency *hold-down* timers are used to insert a minimal time interval between concurrent updates (from now on we will denote this interval as *hd*) These update policies can be made more sensitive to changes in the utilisation when it approaches the link capacity or the sensitivity can be the same across the range of values of link utilisation Hence we further divide these policies into

*equal density utilisation change policy* – advertises change in the link utilisation with an equal density for high and low loaded links

- *class based policy* – it partitions the link bandwidth into classes of equal size and every time a boundary between classes is crossed a new link state advertisement is generated

*increasing density utilisation change policy* – advertises change in the link utilisation with a density increasing with link utilisation Such an approach results in less frequent updates for lightly loaded links This is based on the assumption that a slight increase (decrease) in the link load for a highly utilised link can block (allow acceptance of) the new request When the link utilisation is low this rarely arises situation

- *threshold based policy* – advertises a new link state whenever the magnitude of the link state change exceeds some predefined threshold The magnitude of the link state change decreases with the link utilisation The following notation is used $u_l$ and $b_l$ are the utilisation and available bandwidth of the link at the time of the last update, $u_c$ and $b_c$ are the current link utilisation and the current available bandwidth, $tr$ is the threshold is expressed as a percentage (this is the smallest magnitude of change which will be advertised) A link state update will be generated only when

$$\frac{\mid u_l - u_c \mid}{1 - u_l} * 100 \geq tr,$$

or equivalently when

$$\frac{| b_l - b_c |}{b_l} * 100 \geq tr$$

This update policy exploits an hysteresis technique, which often successfully helps to reduce communication overhead and oscillations [80, 113]

- *class based policy* – it partitions the link bandwidth into classes with size decreasing with increasing link utilisation – every time a boundary between classes is crossed a new link state advertisement is generated

In general, timer-based and utilisation change based update policies can coexist in the network So the new state advertisements are triggered upon significant changes in available bandwidth (a hold-down timer is applied that enforces a minimum spacing between such updates) and also at the end of every refresh period (so providing an upper bound on the time interval between concurrent updates)

We do not assume, or model, any particular technique for distributing state information in the network, two possibilities are flooding (as in PNNI and OSPF) or broadcasting via a spanning tree

## 3.5 Performance metrics

In order to evaluate the performance and costs of QoS routing, we have selected a set of metrics

***call blocking rate*** – defined as the number of rejected requests, divided by the number of all requests,

***bandwidth blocking rate*** – defined as the bandwidth of rejected requests, divided by the bandwidth requested by all incoming requests,

***update overhead*** – defined as the number of link state updates generated per second. [2]

***relative average path length*** – defined as the average path length of accepted requests, divided by $h'$ (the average min-hop path distance between nodes, calculated over all source-destination pairs). This allows us to check whether a routing algorithm uses resources excessively (ratio $> 1$) or conserves resources (ratio $\leq 1$);

***variation in link utilisation*** – which is the average variation in utilisation of all network links. If its value is small the load is spread well over all links, and if it is high the routing algorithm is likely to produce points of congestion, while there are still unused resources.

The call blocking rate and bandwidth blocking rate can be classified as performance metrics. The update overhead represents the overhead metric. And the relative average path length and variation in link utilisation represent the behavioural metrics.

## 3.6 Summary

In this chapter we have presented a simulation model for link state QoS routing algorithms. The model is comprised of: routing algorithms which select paths based on state information; a variety of network topologies capturing the diversity of the Internet; traffic models generating various traffic patterns; update policies controlling the advertisements of link states; and performance metrics used to evaluate the behaviour of routing algorithms. The complete set of parameters

[2]While it might be appealing to express this instead as bytes per second or as a fraction of network capacity, the actual overhead is dependent on the link speed, the routing protocol, and the mechanisms for exchanging link state. Since we only focus on the routing algorithms, and not on the mechanisms for exchanging link state information, the number of link state messages is the most suitable.

used in our simulations is presented in Table 3 2 Our model is flow-based and it captures the nature of connection-oriented networks such as MPLS or ATM

This model combines various models used by other researchers to provide a realistic simulation environment Moreover, it can be easily configured to analyse the performance of QoS routing algorithms

**Table 3.2:** *Set of parameters used in simulation model*

| | | |
|---|---|---|
| **INPUT** | | |
| **Routing algorithm** | | |
| **WSP** | | |
| **EXP** | | |
| **ALCFRA** | | |
| **CAR** | | |
| **Network** | | |
| **Topology** | ISP topology<br>Tree topology<br>Low connectivity topology<br>High connectivity topology<br>Two-level hierarchical topology | |
| **Link capacity** | | C |
| **Traffic** | | |
| **Load** | | $\rho$ |
| **Interarrival times** | exponentiall<br>Weibull | mean<br>mean<br>shape parameter |
| **Holding times** | exponentiall<br>Pareto | mean<br>mean<br>shape parameter |
| **Requested bandwidth** | | min<br>max |
| **Update Policy** | | |
| **Policy** | threshold-based<br>class-base | tr<br>class size |
| **Hold-down timer** | | hd |
| **OUTPUT** | | |
| **Performance metrics** | | |
| **Call blocking rate** | | |
| **Bandwidth blocking rate** | | |
| **Update overhead** | | |
| **Relative average path length** | | |
| **Variation in link utilisation** | | |
| **Control Parameter(s)** | | |
| **Load** | | from ... to ... |
| **Hold-down timer** | | from ... to ... |
| **Links Number** | | from ... to ... |

# CHAPTER 4

# QoS Routing: Theory Vs. Practice

There is a wide set of proposed approaches to QoS routing, but our investigations suggest that researchers usually take one of two contradictory approaches either resource conserving (RC) or load distributing (LD) These approaches have been introduced in Sections 2 5 1 and 2 5 2

Such a situation is due to the fact that LD algorithms are theoretically optimal So some researchers like to use them to achieve efficient resource utilisation However LD algorithms often results in oscillatory behaviour which leads to poor network utilisation So many researchers prefer the more stable mechanism of the RC approach These algorithms do not try to perform in a theoretically optimal way but feature stable performance In Table 4 1 we show the main differences between these two approaches

In this chapter we analyse the stability of the LD approach using such tools as estimation error, network response and link cost sensitivity We want to check how unstable LD algorithms really are compared with RC algorithms Therefore we evaluate the LD and RC approaches in a realistic scenario We simulate the behaviour of these mechanisms in an environment very close to real networks, i e ,

**Table 4.1:** *Load Distribution vs. Resource Conservation*

| | **Load distribution approach (LD)** | **Resource conservation approach (RC)** |
|---|---|---|
| *primary link cost metric* | depends on link utilisation | is a constant |
| *resource utilisation* | all network links are usable | only links belonging to the set of min-hop paths are usable |
| *resource conservation* | not present | present |
| *vulnerability to inaccuracy of network state information* | high (link cost is a function of link utilisation) | low (link cost is constant) |
| *traffic fluctuations can occur* | over all links | over the set of min-hop paths |

in the presence of inaccurate information. In such an environment traffic oscillations are likely to occur and this makes it very difficult to achieve the theoretical optimum performance. This allows us to check whether theoretically optimal solutions are applicable in real networks.

## 4.1 Instability of the Load Distributing approach

Load balancing routing algorithms perform well in a network environment where traffic load is light [98] and network conditions change slowly. Such routing algorithms are able to balance the network load over all possible paths and adjust routing decisions in response to traffic changes. Therefore in the presence of congestion such algorithms can redirect traffic away from the overloaded paths.

Unfortunately the LD routing algorithms which attempt to adapt to traffic changes (using information about available bandwidth as considered in this thesis) frequently exhibit instability [28, 29]. Traffic oscillations are most likely to occur when the costs of different paths vary widely. Therefore the paths which have a lower cost may attract the majority of the incoming traffic and become congested while the paths which reported a high cost may become idle. When rerouting of existing flows is permitted such fluctuations are magnified because when the original route between two nodes becomes congested, all the traffic to that destination can be shifted from the original route to an alternate route This may cause congestion on the alternate route Traffic may have to be shifted again [72] The imprecision of the network state information increases the likelihood that traffic will fluctuate, because appropriate link costs are advertised infrequently Thus, during path selection process routers do not know exactly which path is best and which provides sufficient resources This leads to poor performance of the QoS routing algorithm [8] and reduces the network throughput

To measure the stability of the system we use three factors *estimation error, link cost sensitivity,* and *network response*

### 4.1 1 Estimation Error

Link state QoS routing methods select paths based on state information which is usually out-of-date, because providing accurate and timely value for link cost is extremely hard Therefore the link cost values used by routing algorithms are estimates rather than the actual values The difference between the estimated value and the accurate measured value is called the *estimation error* [148, 149]

When the estimation error increases, the estimated distances which the route computation is based on become less valid and therefore the performance of the route selection deteriorates So the estimation error can be used as a measure of routing accuracy High estimation error also increases traffic fluctuations

Suppose that $u^l(e)$ is the last advertised utilisation value for link $e$, and that $u^c(e)$ is the current link utilisation of link $e$, then the estimation error $\Lambda$ of link cost when the LD approach is used is given by

$$\Lambda = \mid cost^c_{EXP}(e) - cost^l_{EXP}(e) \mid = \frac{(MAX - MIN)}{A-1} \mid A^{u^c(e)} - A^{u^l(e)} \mid,$$

Knowing that traffic traversing the link can vary, we expect that the current link utilisation $u^c(e)$ can be different from the last advertised value $u^l(e)$ We denote thıs change by $\epsilon$ and we express ıt as

$$\epsilon = u^c(e) - u^l(e),$$

thus, the estımatıon error becomes

$$\Lambda = \frac{(MAX - MIN)}{A-1} \mid A^{u^c(e)} - A^{u^l(e)} \mid = \frac{(MAX - MIN)}{A-1} \mid A^{u^l(e)+\epsilon} - A^{u^l(e)} \mid,$$

and fınally

$$\Lambda = \frac{(MAX - MIN)}{A-1} A^{u^l(e)} \mid A^{\epsilon} - 1 \mid \quad (4\ 1)$$

An effectıve routıng ıs performed only ıf there ıs some correlatıon between the reported values and those actually experıenced after routıng For the pure LD approach, the correlatıon between successıve advertısed lınk costs ıs hıgh when the network ıs lıghtly loaded However, the estımatıon of lınk costs deterıorates under heavy traffıc loads, sınce the estımatıon error ıs proportıonal to $u^l(e)$ and $\epsilon$ ($MAX$, $MIN$, and $A$ beıng constant) Thus for heavy loads the estımatıon error ıs hıgh resultıng ın a poor route selectıon process

### 4.1.2 Lınk Cost Sensıtıvıty

We defıne lınk cost sensıtıvıty as the relatıve change ın the value of the lınk cost functıon due to a change ın the lınk utılısatıon

$$\text{Lınk Cost Sensıtıvıty} = \frac{\%\text{ change ın } cost(e)}{\%\text{ change ın } u(e)} \quad (4\ 2)$$

Thus for a small varıatıon ın lınk utılısatıon the sensıtıvıty of the cost functıon of lınk $e$ can be wrıtten as

$$S(e) = \frac{\frac{\partial cost(e)}{cost(e)}}{\frac{\partial u(e)}{u(e)}} = \frac{u(e)}{cost(e)} \frac{\partial cost(e)}{\partial u(e)} \quad (4\ 3)$$

The sensıtıvıty functıon above consıders only a sıngle lınk Ideally we should consider a network sensitivity formula, which measures the network-wide stability of the routing algorithm in the presence of load variations on individual links. Such a formula would be extremely complex. However, since all network links use the same formula for link cost, high sensitivity of the link cost to small transient changes in the link utilisation may be expected to trigger traffic oscillations.

Let us now examine the sensitivity of the cost function of link $e$ when the LD approach is used.

$$S_{EXP}(e) = \frac{u(e)}{cost_{EXP}(e)} \frac{\partial cost_{EXP}(e)}{\partial u(e)} = \frac{u(e)A^{u(e)} \ln A}{A^{u(e)} - 1} \tag{4.4}$$

An example of the link cost sensitivity of the LD approach (where $MAX = 1$, $MIN = 0$ and $A = 100$) is shown in Figure 4.1.

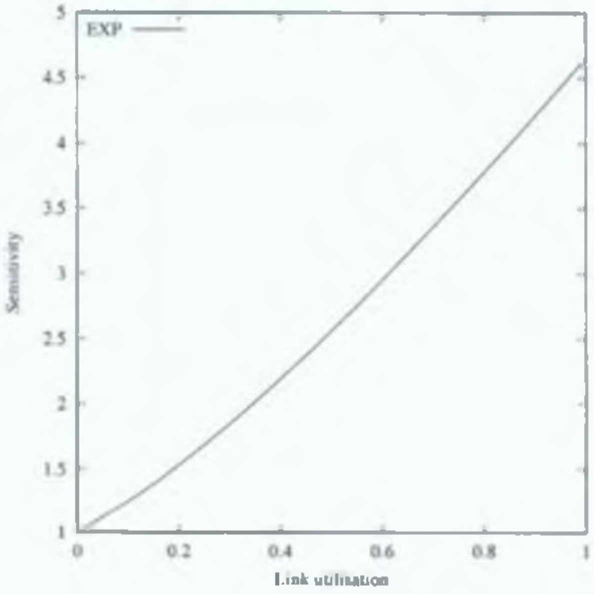


**Figure 4.1:** *Link Cost Sensitivity of the LD approach*

As we can observe for the EXP algorithm (representing the LD approach and introduced in Section 3.1) the link cost sensitivity increases with the link utilisation. It is most sensitive when the utilisation approaches the link bandwidth. For highly utilised links small transient changes in the link utilisation can result in major changes in the link cost. Therefore link fluctuations are more likely to occur for highly loaded networks when LD approach is used.

### 4.1.3 Network response

No theoretical method is available for analysing the effectiveness of the feedback of link state advertisements in reducing the influence of the variation in link cost value on the stability of the system performance. However, we can examine the mechanism of traffic fluctuations using a graphical method. It reflects the probability of occurrence of traffic oscillations and the method is based on the *network response* [85].

To describe the meaning of network response we define the route selection process as the following cycle:

1. The Routing algorithm selects routes consisting of a set of links, based on link costs, network topology and route requests.

2. The sum of the traffic on each link gives the link utilisation.

3. The link utilisation is converted into a cost which is advertised to other nodes in the network.

4. Next this link cost is used to calculate new routes, so the cycle repeats itself.

Figure 3.2 shows the link cost function used by the LD algorithms, and it represents the mapping of the link utilisation into link cost. The *network response* [85] represents the opposite mapping: from cost to utilisation. This mapping cycle of the link cost to the link utilisation and the link utilisation to the link cost is shown in Figure 4.2. Thus the network response shows how the link utilisation will change in the next period of time due to advertising a new cost. If the new computed cost is the same as the old cost the link is at equilibrium. The whole network is at equilibrium, when all links are at equilibrium.

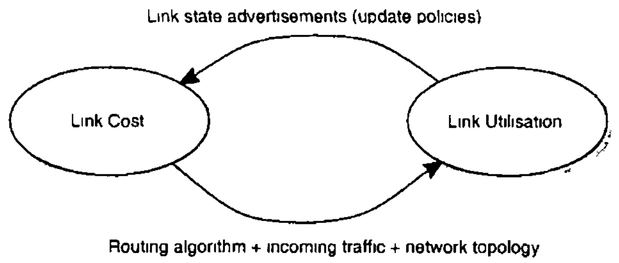

Link state advertisements (update policies)
Link Cost
Link Utilisation
Routing algorithm + incoming traffic + network topology

**Figure 4 2** *Cost ⟷ utilisation mapping cycle*

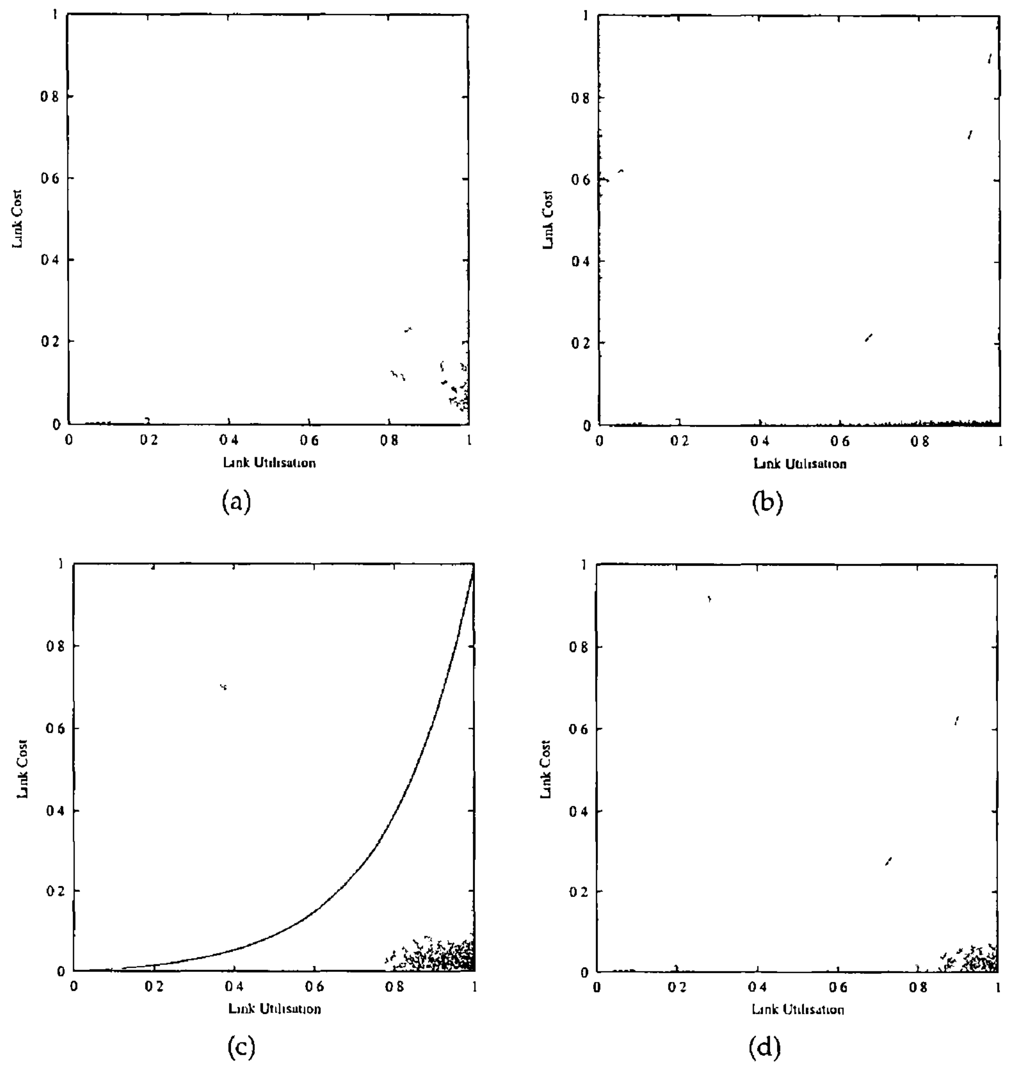


**Figure 4 3** *Examples of network response for a highly utilised link*

The network response together with the link cost function allow the dynamic behaviour of the system to be analysed If we place these two functions on a single graph we can track how a change in the link utilisation affects the link cost, and how it in turn affects the link utilisation in the next update interval

Therefore, we can check whether the system ıs prone to oscıllatıons by vısual ınspectıon

The authors of [85] ın obtaınıng the network response assumed that all lınk cost values were fıxed and only the cost of a sıngle lınk was changed to observe how ıt ınfluences the utılısatıon We consıder a more general case, wıth the ınfluence of all lınk costs on the utılısatıon of a consıdered lınk We focus on the probabılıty of occurrence of traffıc fluctuatıons due to the ınteractıon between lınk cost functıon and network response Also we try to generalıse our dıscussıon to varıous topologıes

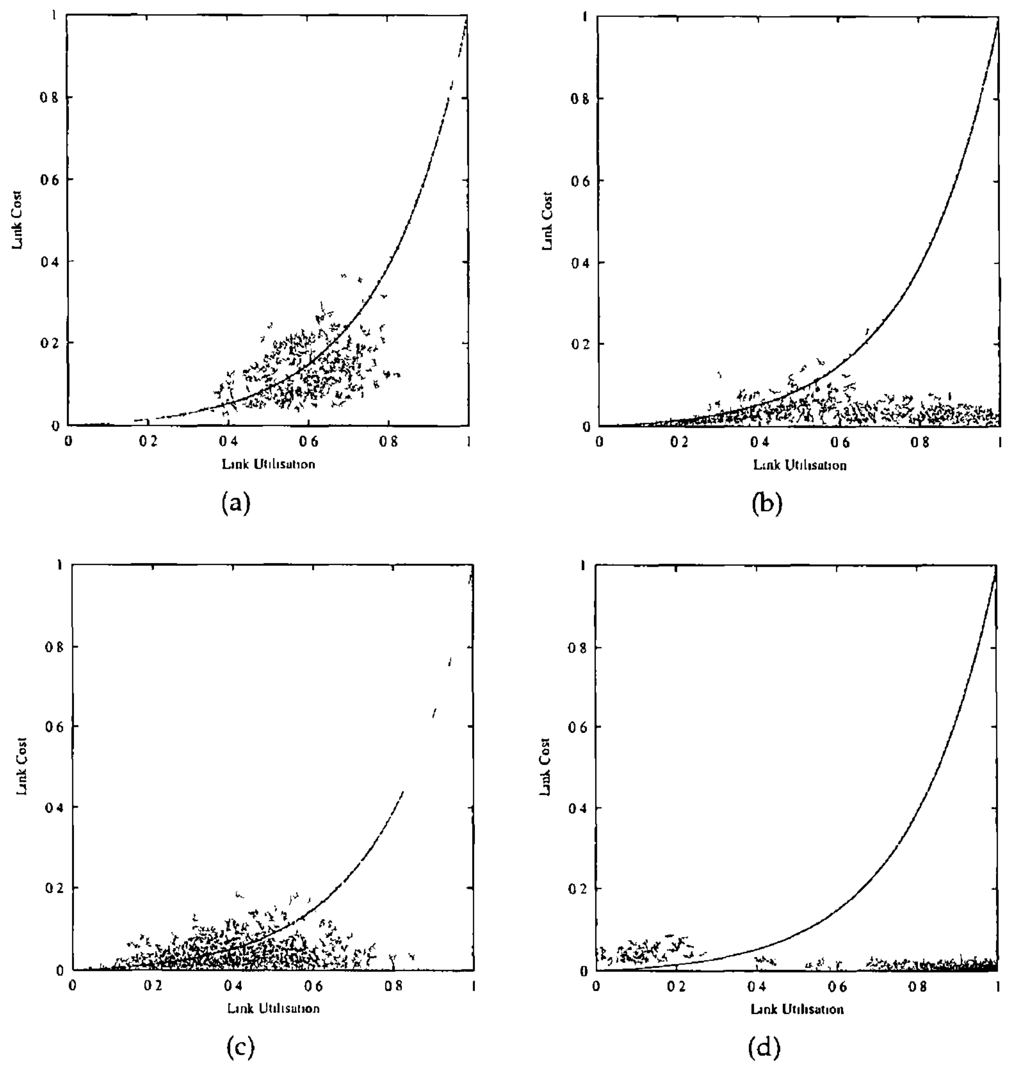


**Fıgure 4 4** *Examples of network response for a lıghtly utılısed lınk*

Our goal ıs to observe how susceptıble routıng ıs to traffıc fluctuatıons, and so we want to observe if it is easy for the link to move away from its equilibrium point. Thus we observe a situation when the average long term link utilisation is fixed, and we only observe transient fluctuations in link utilisation, which occur due to the establishing and closing of connections. In Figures 4.3 and 4.4 we present a few examples of network response observed for links with high and low mean utilisations, respectively. The dots in Figures 4.3 and 4.4 represent the observed network response and the line is the link cost function.

It is evident from the presented examples that some links respond to a small change in the link utilisation in a controlled manner, so that the areas of network response are close to the average link utilisation, as shown in Figure 4.5. While in other areas the link utilisation is oscillatory, and the highest traffic fluctuations are observed when it oscillates between its maximum and minimum values, so the areas of network response fall into regions shown in Figure 4.6.

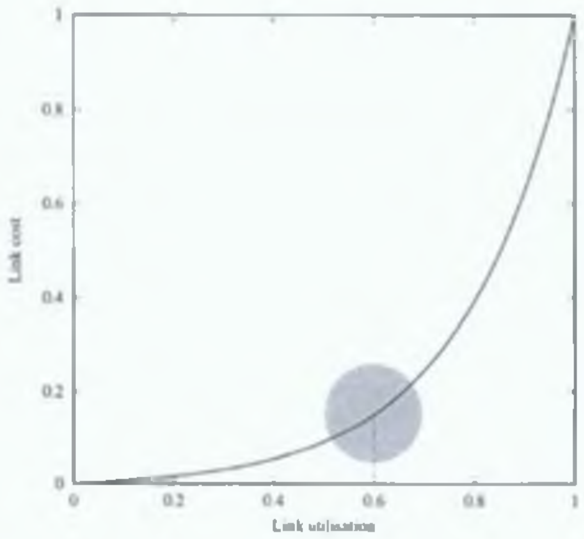


**Figure 4.5:** *Stable area of network response*

However, in most cases we can observe that the network response is based on two rules. The link utilisation increases after advertising a link cost lower than at equilibrium and decreases after advertising a link cost higher than at equilibrium. Such behaviour can be observed with different levels of magnitude and this is caused by various network topologies, traffic patterns and update policies. So

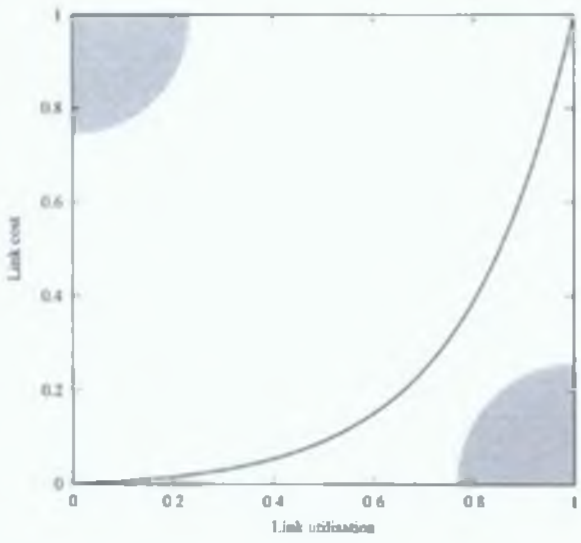
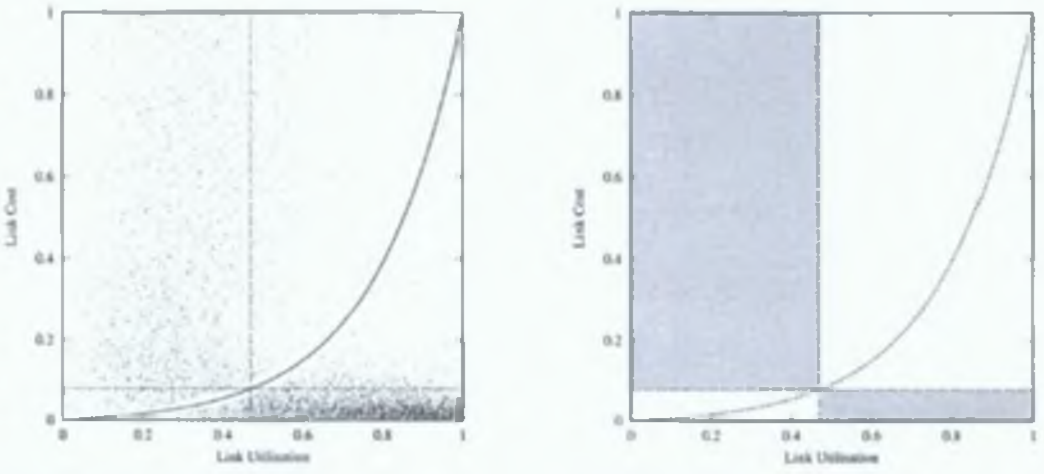


**Figure 4.6:** *Unstable areas of network response*

to generalise our discussion we define typical areas of network response, which occupy regions of increased load after advertising lower cost than at equilibrium and decreased load after advertising a link cost higher than at equilibrium. This is shown in Figure 4.7.



(a) Observed areas of network response (b) Typical areas of network response

**Figure 4.7:** *Observed and typical areas of network response*

The points of network response, which have the potential to trigger traffic fluctuations, are located at the top-left and bottom-right corners of the graph shown in Figure 4.6. These points determine the worst-case response. The best case is when the network response corresponds to the link cost function, i.e., it is situated on the curve shown in Figure 4.6. We make the assumption that by drawing a line between the best case and worst cases we obtain an approximation to the average network response, as shown in Figure 4.8.

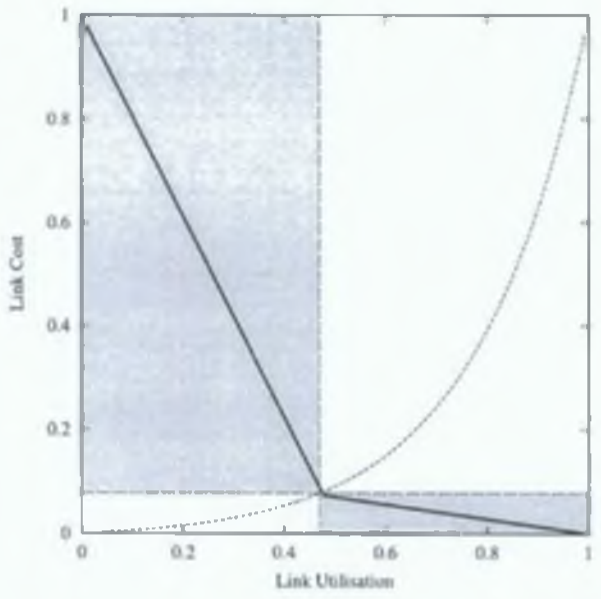


**Figure 4.8:** *Average network response*

In the rest of our discussion about network response we presume that using the link cost function and the average network response we can observe system behaviour. This is a simplifying assumption and is intuitively appealing. Although is has no formal justification, it provides interesting insights into the problem of route selection instability caused by the shape of link cost function. Knowing the link cost, we can find the resulting network response (utilisation). Knowing the network response, we can find the corresponding link cost. Hence we can draw trajectories and approximately analyse the dynamic behaviour of the system.

Such trajectories can be traced by starting at a certain utilisation level close to the equilibrium point and finding the corresponding cost on the link utilisation function. This advertised cost value will result in a new traffic level which can be found from the average network response function. The dynamic behaviour can be found by repeating this process.

This allows us to check how the link utilisation changes due to a small, transient change in the link cost, and whether or not it triggers traffic fluctuations Such behaviour can be better explained using examples We consider two such cases below, one where the link utilisation is high, and another where it is low

#### 4 1 3 1 Link with high utilisation

First we consider a highly utilised link, where the currently observed link utilisation is around 90% as shown in Figure 4 9

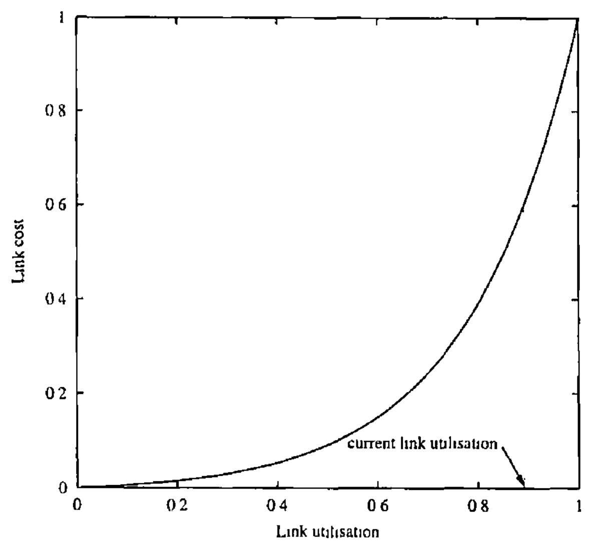


**Figure 4 9** *Link cost for a link with high utilisation*

In Figure 4 10 we mark the typical areas of network response

For such typical network response areas we can define an *average network response* as shown in Figure 4 11 This represents an average reaction to a change in the link cost, so this is what we can expect after advertising a new link cost

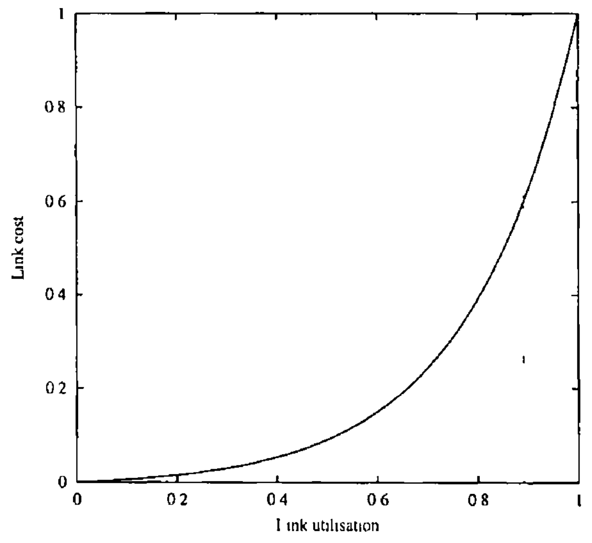


**Figure 4 10** *Typical areas of network response for a link with high utilisation*

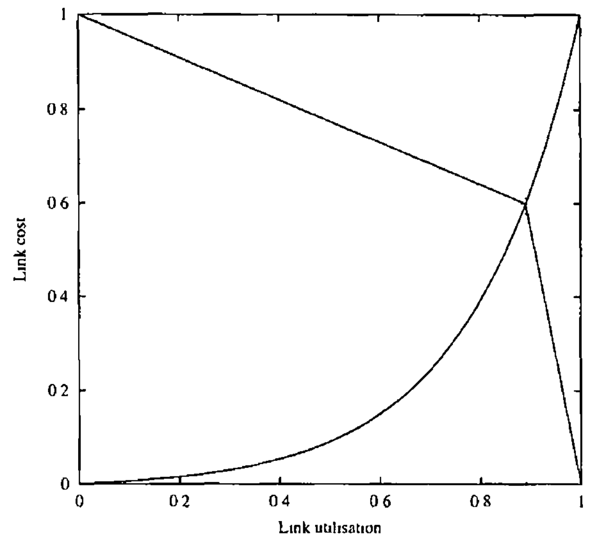


**Figure 4 11** *Average network response for a link with high utilisation*

When a link is highly utilised, a small change in link utilisation can produce traffic fluctuations, as shown in Figure 4 12 An increase in the link cost leads to a decrease in the link utilisation and so to a decrease in the link cost in the next update interval This leads to an increase in the link utilisation and thus in the link cost in the subsequent time interval, etc Therefore severe oscillations are observed

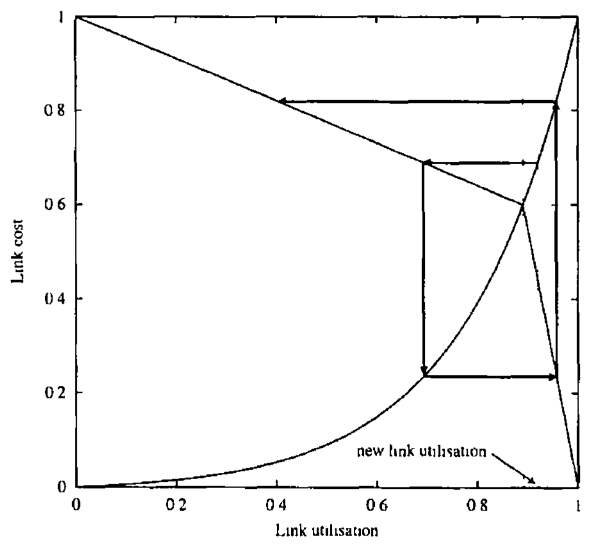


**Figure 4 12** *Unstable link cost⟷utilisation trajectory for a link with high utilisation*

### 4 1 3 2 Link with low utilisation

Let us consider a link with low utilisation, where the currently observed link utilisation is around 40% as shown in Figure 4 13 The typical network response areas and average network response are shown in Figures 4 14 and 4 15 respectively

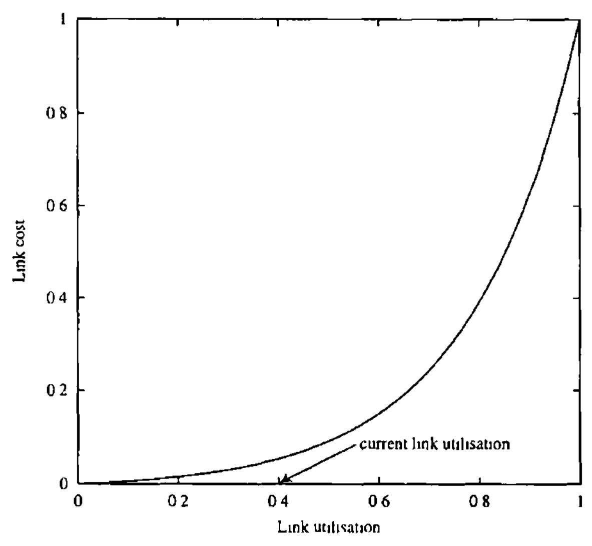


**Figure 4 13** *Link cost for a link with low utilisation*

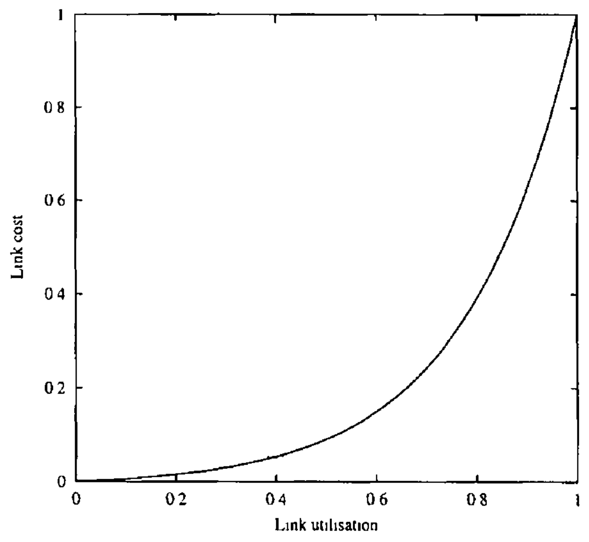


**Fıgure 4 14** *Typıcal areas of network response for a lınk wıth low utılısatıon*

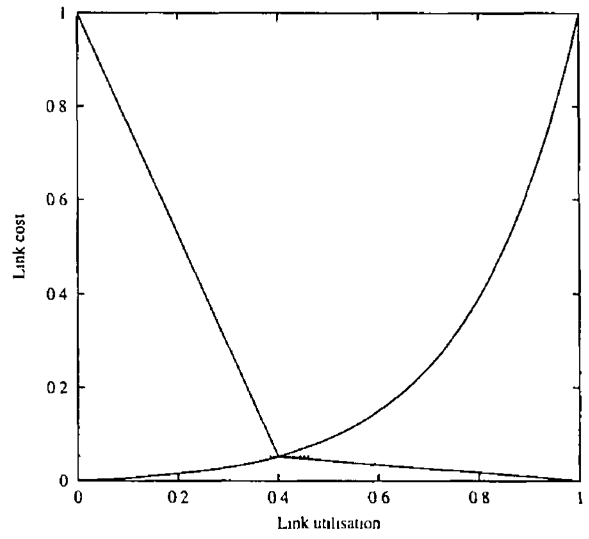


**Fıgure 4 15** *Average network response for a lınk wıth low utılısatıon*

In thıs case ıf we observe the dynamıc behavıour of the lınk as shown ın Fıgure 4 12, we can see that small changes ın lınk utılısatıon do not trıgger traffıc fluctuatıons, and small oscıllatıons are damped Thıs verıfıes the observatıon that the LD method performs well ın the network envıronment where traffıc load ıs lıght Thus only transıent traffıc oscıllatıons are observed, whıch are moreover easıly damped

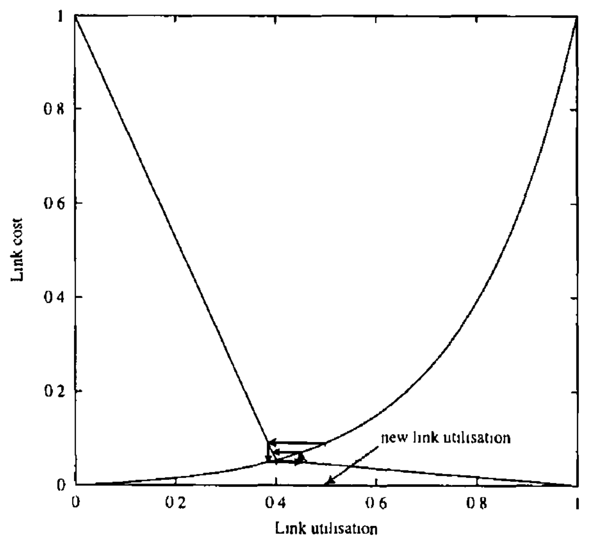


**Figure 4 16** *Stable link cost⟷utilisation trajectory for a link with low utilisation*

### 4 1.4 Instability of the LD approach: Summary

As we observed in previous sections, if link utilisation is low, the link is likely to converge to the equilibrium If the utilisation is high, the system will probably diverge and oscillate The equilibrium at highly utilised links can be considered as meta-stable because a slight perturbation can knock the system off its equilibrium and into the realm of instability

A rigourous analysis of the network response would be extremely complicated The few examples presented here give a general idea about how easily traffic oscillations are triggered when the link utilisation approaches the total bandwidth

## 4.2 Load Distribution (LD) vs. Resource Conservation (RC)

Load distribution (LD) and resource conservation (RC), are conflicting goals of QoS routing They also represent a conflict between theory and practice

In general the RC approach is more stable than the LD approach LD algorithms utilise all network resources so as to avoid congestion, which often results in the network traffic fluctuating over all links RC algorithms are restricted to the set of minimum-hop paths, and so can produce such oscillations only on links belonging to these paths

It is a commonly held view, that traffic distribution pays off when the network load is low, while conserving resources performs better at high loads However we want to check if increasing load is the only factor which causes the performance of the LD algorithms to deteriorate Intuitively it is clear that the network topology strongly influences the occurrence of traffic oscillations Moreover some researchers coincidentally or otherwise simulate topologies which give the best results for their chosen approach To be precise, researchers supporting the LD approach use networks which are sparsely connected [65, 66, 82], while researchers advocating the RC approach use networks with rich connectivity [120, 131, 133] Therefore the influence of the network topology on the routing performance needs to be verified This is the purpose of this section We also want check whether the observations made by other researchers about the performance of the LD and RC approaches are valid for a wide range of network topologies

### 4 2.1 Performance evaluation of LD and RC approaches

To compare the performance of the above two approaches to QoS routing we study their use on a wide variety of network topologies In the rest of this section the LD and RC approaches are represented by the EXP algorithm and WSP algorithm respectively (these algorithms are described in Section 3 1)

#### 4.2.1.1 Simulation model

Full details of the model used in our simulations are provided in Chapter 3 However, in this section we outline elements of this model which apply specifically to this experiment Chapter 3 describes the details of link state QoS routing algorithms and provides a comprehensive model for such an architecture to be used in virtual-circuit networks It is also consistent with models used by other researchers (e g , [10, 133, 160])

Our goal is to show how routing algorithm performance is affected by the level of network connectivity Thus we require a network topology whose connectivity can be adjusted in a controlled manner To achieve this, we use a base network with $N$ nodes and $L$ links and add links randomly until the required connectivity is achieved As a base network we use the topology shown in Figure 3 3(b), which has $N = 21$ nodes and $L = 20$ links (the minimal level of connectivity corresponding to $L = N - 1$), so the average node degree is equal to $2L/N = 1.9$ We add links randomly to our base network until the number of links is $L = 80$ which gives an average node degree of 7 6

The network connectivity is an important factor when the LD approach is evaluated, while when the RC approach is used a more meaningful parameter is the number of min-hop paths, so for each connectivity level we also present the average number of min-hop paths between any source and destination pair

We extend the base network by adding links until rich connectivity is achieved It would also be possible to remove links from a highly connected network to obtain lower connectivity The mechanism used to deal with broken connections could have a significant impact on overall performance increasing the variability of results obtained So we decided that adding links to a base network of low connectivity is the better approach

Adding one or more links to a network can result in different routing performance, depending on the placement of the newly added links Therefore this process needs be repeated several times to get a stable and uniform result In our experiment we create 200 different network topologies with the same connectivity level, and average their performance All the average results are obtained with a 95% confidence interval We also assume that links are bidirectional, each with identical capacity $C$ (in our simulations we use DS-3 links with capacity $C = 45Mbps$) and the offered traffic has a uniform pattern, as presented in Section 3 3

There are two essential factors of network topology that determine which of the two approaches performs better, the level of network connectivity and the number of min-hop paths between any source and destination pair The LD approach utilises all links, trying to spread the traffic over all network So the number of links, represented by the level of connectivity, has an important impact on the performance of the LD approach The performance of the RC approach depends more on the number of min-hop paths than the connectivity level, because it requires the presence of at least a few min-hop paths in order to perform better than static routing (which ignores link utilisation) That is why on all graphs in this experiment we present both the level of network connectivity and the number of min-hop paths

### 4 2 1 2 LD approach vs RC approach

The first step of our experiment compares both algorithms under a light load, $\rho = 0\ 3$ As shown in Figure 4 17 when the load is light and the state of the information is up-to-date ($hd = 1sec$), LD gives the better performance Similar results occur even if the level of inaccuracy in network state information is much higher, for example when $hd = 40sec$ as shown in Figure 4 18 This confirms the findings presented in [98] and can be explained as follows using extra resources, in the form of longer paths, has negligible importance when the network load is low, because there are still sufficient uncommitted resources to accommodate future connections Selecting shortest-path routes cannot fully exploit all network resources, and produces a higher blocking rate This conclusion remains valid over a wide spectrum of network connectivity (average node degree from 1 9 to 7 6)

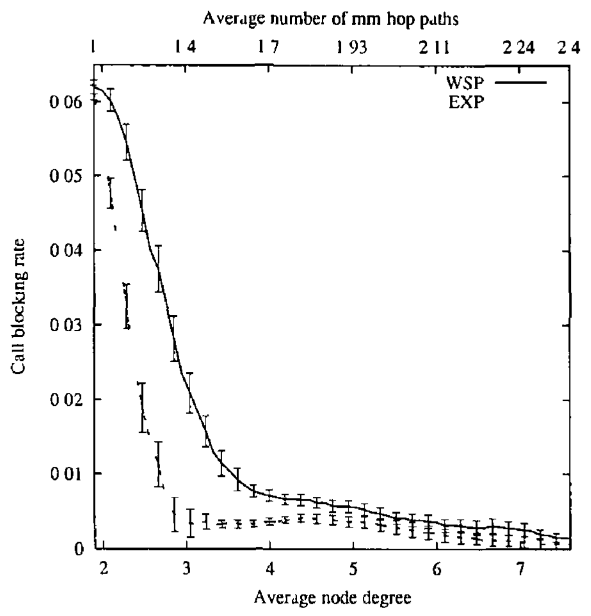


**Figure 4 17** *Call blocking probability of WSP and EXP versus increasing node degree,* $\rho = 0\ 3$, $hd = 1sec$

The situation changes if the network load is high As shown in Figure 4 19 if accurate state information is provided ($hd = 1sec$), LD makes better use of network resources for networks with node degree less than 6 However if inaccuracy is higher, for example when $hd = 20sec$ or $40sec$ as shown in Figure 4 20 and Figure 4 21 respectively, the RC approach performs well even for less connected networks (for node degree $> 4\ 5$ if $hd = 20sec$ and $> 3\ 5$ if $hd = 40sec$) This suggests that load balancing approaches cannot operate well in the presence of very inaccurate state information The RC approach is more tolerant of imprecise state information, and for a highly connected network this gives a huge advantage over the LD approach

Nevertheless there are networks with low connectivity for which only load balancing approaches can improve on static shortest path routing

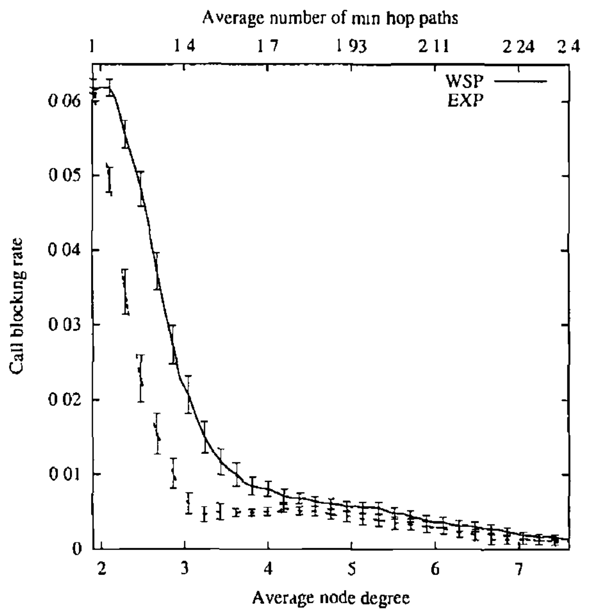


**Figure 4 18** *Call blockmg probabılıty of WSP and EXP versus increasing node degree, $\rho = 0\ 3$, $hd = 40sec$*

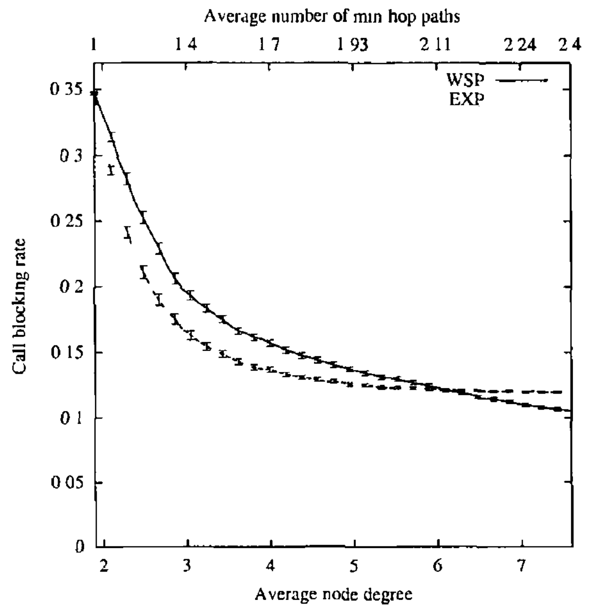


**Figure 4 19** *Call blockmg probabılıty of WSP and EXP versus increasing node degree, $\rho = 0\ 8$, $hd = 1sec$*

These results show that network topology, traffic load and accuracy of state ınformatıon play an ımportant role ın decıdıng whıch routıng strategy performs

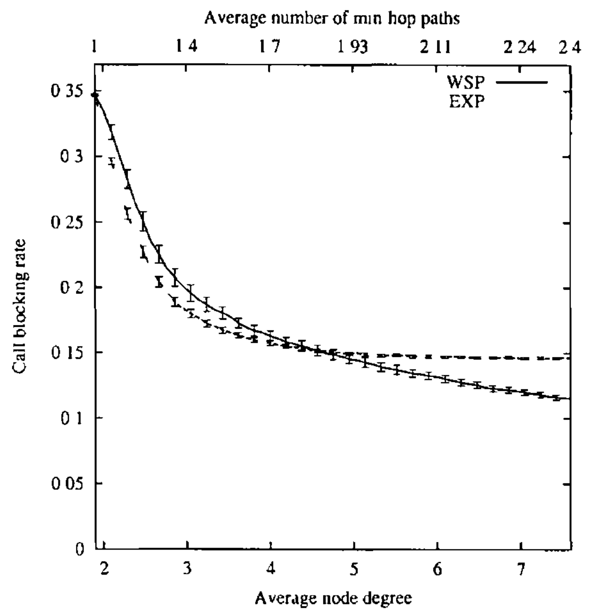


**Figure 4 20** *Call blocking probability of WSP and EXP versus increasing node degree, $\rho = 0\ 8$, $hd = 20sec$*

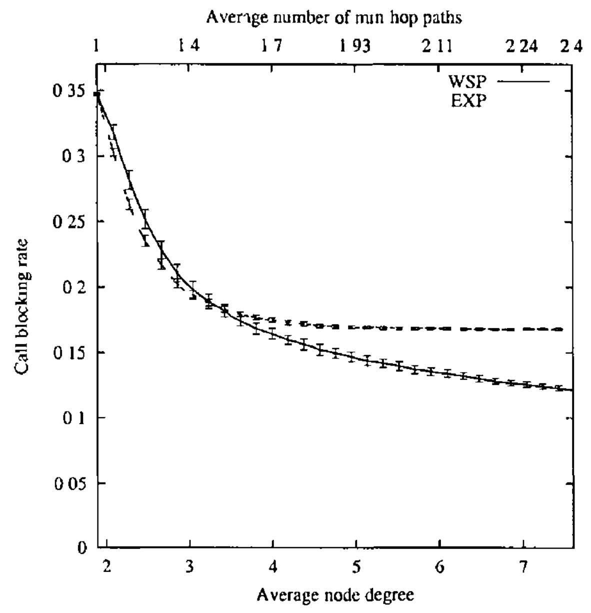


**Figure 4 21** *Call blocking probability of WSP and EXP versus increasing node degree, $\rho = 0\ 8$, $hd = 40sec$*

better Hence none of these three factors can be neglected in the evaluation of routing mechanisms

## 4.3 Summary

In this chapter we have investigated the stability issues of the LD approach to routing Such algorithms are likely to exhibit instability when the link load approaches the total bandwidth

From our comprehensive comparison of LD and RC approaches we have made the following observations

- For networks with low connectivity the LD approach outperforms the approach which limits hop count, because it uses scarce resources in a more efficient way
- RC algorithms typically perform better for highly connected networks, when there are at least a few min-hop paths between any source and destination
- For lightly loaded networks the LD approach seems to be a much better solution than RC
- For highly loaded networks the RC approach offers better performance, especially in the presence of very imprecise state information

Therefore in general we can say that all three factors network topology, traffic load and accuracy of state information are necessary to consider the benefits of the RC and LD approaches

Our findings show the dangers of drawing conclusions about the performance of a routing algorithm on the basis of results obtained using a limited set of network topologies Because of the wide diversity of network topologies constituting the current Internet, routing algorithms should be tested over a wide spectrum of connectivity Only such an approach allows general conclusions to be drawn about the performance of the algorithm

In summary these observations suggest that in order to allow QoS routing to perform well over a wide range of network topologies it cannot be restricted only to the set of mın-hop paths In the next chapter we propose an algorıthm whıch ınherıts propertıes of both the RC and LD mechanısms to achıeve an effıcıent usage of the resources and low vulnerabılıty to ımprecıse network state ınformatıon

# CHAPTER 5

# Methods for Damping Traffic Fluctuations

In this chapter we propose methods for damping traffic fluctuations in virtual circuit networks Fears concerning traffic fluctuations and the resulting unstable performance of QoS routing have militated against the deployment of these algorithms in operational networks Therefore we need effective methods to tackle this problem

In Section 2 2 4 we describe the methods used to damp traffic oscillations in datagram networks However, the solutions proposed for datagram networks [29] cannot be directly applied to the virtual circuit environment considered in this thesis

Methods of damping traffic oscillations for virtual circuit networks have attracted scant attention Therefore we propose two novel algorithms

- **ALCFRA** – Adaptive Link Cost Function Routing Algorithm, which improves the stability of the LD approach and is intended for use in networks with sparse connectivity

- **CAR** – Connectivity Aware Routing, which is a hybrid of the LD and RC approaches and is designed to work well in most other network topologies

We emulate the behaviour of these mechanisms in an environment very close to that of real networks, by modelling the presence of inaccurate information In such an environment, traffic oscillations are likely to occur and this makes it very difficult to achieve theoretical optimum performance The route selection improvements presented in this chapter not only stabilise the link state QoS routing methods, but also achieve an excellent performance over a variety of network topologies

## 5.1 ALCFRA

The idea of stabilising QoS routing by averaging the link cost metric value over a period of time covering more than one link state update (see Section 2 2 4), was originally proposed for datagram networks [29] In such networks, link utilisation can change very rapidly, so by averaging we can effectively stabilise its value [29]

In virtual circuit networks the speed of network response to congestion is much slower and depends on connection arrival and departure processes as well as on the frequency of link state updates [29] Furthermore, in virtual circuit networks featuring resource reservation, instead of measuring the link utilisation we can assume that it is proportional to the amount of reserved bandwidth This already stabilises the link cost metric value, and additional averaging does not improve it further In datagram networks the process of averaging stabilises the link cost value, but in virtual circuit networks it only slows down the response to congestion of pure load balancing algorithms (as we show in Section 5 1 5)

By averaging the link cost value, the long term link utilisation is provided rather than its current value When the characteristics of the traffic traversing the network change infrequently, the long term utilisation can convey useful information. A novel method of employing such information is used in ALCFRA (Adaptive Link Cost Function Routing Algorithms). It uses the long term link utilisation to modulate the shape of the link cost function. This method does not reduce the speed of network response to congestion, but makes it less sensitive to slight and transient changes of link state.

The ALCFRA algorithm can thus be used to reduce the traffic fluctuations of the LD approach.

### 5.1.1 The basics of ALCFRA

ALCFRA aims to reduce the effect of traffic oscillations for the LD approach under heavy traffic conditions. It modulates the shape of link cost function used in the LD approach (and given by the Eq. (3.3)) to minimise fluctuations. When the link utilisation is low it uses a convex function as the LD algorithms do, because this does not produce oscillations. However when the link utilisation is high it uses a concave cost function, which does not produce traffic oscillations under heavy load.

The typical network response areas and average network response of the ALCFRA cost function for a highly utilised link are shown in Figure 5.1. We can observe in Figure 5.2 that a small change in the link utilisation in this case results in slight increase in the link cost. Therefore traffic fluctuations are not triggered.

The ALCFRA algorithm does not use two link cost functions, switching between them when traffic conditions change. Since such a mechanism could potentially give rise to traffic oscillations. Instead ALCFRA slowly modulates the link cost function according to the long term link utilisation, so as to achieve stable behaviour. This ensures that transient changes in link load do not produce traffic oscillations and that the algorithms converges to an equilibrium point. A few examples of the modulation of the link cost function are shown in Figure 5.3.

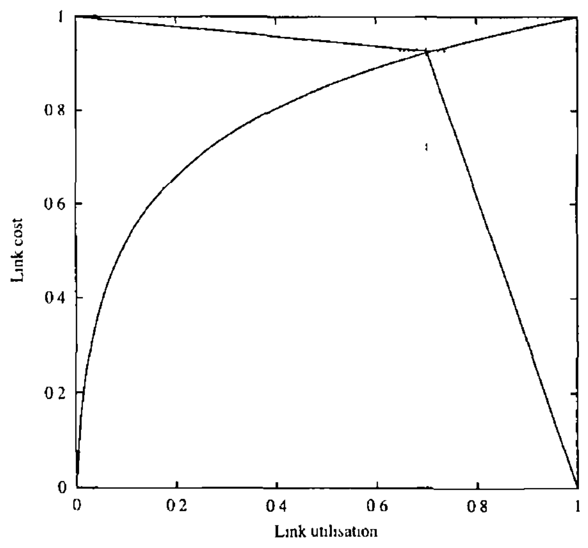


**Fıgure 5 1** *ALCFRA average response to changes ın lınk cost*

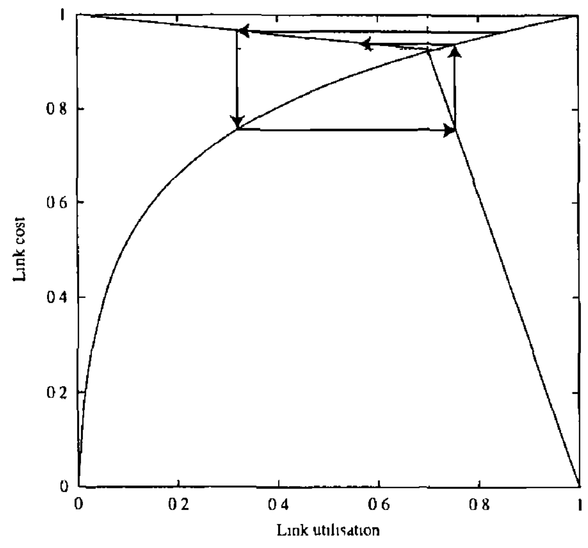


**Fıgure 5 2** *ALCFRA stable lınk cost$\longleftrightarrow$utılısatıon trajectory*

## 5 1 2 Adaptatıon of the lınk cost functıon ın ALCFRA

The ALCFRA algorıthm uses a normalısed exponentıal functıon (gıven by Eq (3 2)) as a base lınk cost metrıc, but modıfıes ıts shape to reflect the long term lınk state utılısatıon The adaptatıon of the lınk cost functıon aıms to prevent sıtuatıons when a small change ın the load for a lınk whıch ıs usually hıghly utılısed would result ın a huge change ın the lınk cost Such adaptatıon ıs done usıng a parameter $a_e$ maıntaıned for each lınk $e$ We assume that the maxımal value of $a_e$ ıs $A$ and

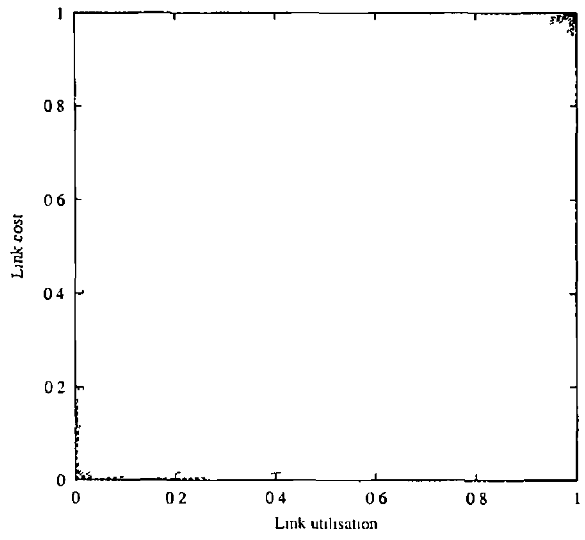


Fıgure 5 3 *ALCFRA examples of modulatıon of the lınk cost functıon*

that $a_e$ can be ıncremented or decremented by some constant value $\Delta$ The value of parameter $a_e$ fluctuates, trackıng the long term lınk utılısatıon, ıt ıs posıtıve when the long term lınk utılısatıon ıs under the predefıned value $u^{tr}$, we call thıs value the threshold utılısatıon (the meanıng of whıch wıll be descrıbed below), and becomes negatıve when the lınk load crosses that value The parameter $a_e$ changes ın the followıng way

$$a_e = \begin{cases} a_e + \Delta, & \text{when } a_e \leq A\left[u^{tr} - u(e)\right]/u^{tr} \\ a_e - \Delta, & \text{otherwise} \end{cases} \tag{5 1}$$

The value of $a_e$ ıs modıfıed as above after every unıt of tıme (ın our sımulatıons the unıt of tıme ıs equal to 1 second) The parameter $a_e$ reflects a long term tendency ın the lınk utılısatıon So the adaptatıon of the base exponentıal functıon ıs performed as follows

$$\mathbf{adapt}(e) = \begin{cases} \frac{a_e^{u(e)}-1}{a_e-1}, & \text{if } a_e > 1 \\ u(e), & \text{if } -1 \leq a_e \leq 1 \\ log_{|a_e|}\Big((\mid a_e \mid -1)u(e) + 1\Big), & \text{if } a_e < -1 \end{cases} \tag{5 2}$$

Finally the link cost function used in ALCFRA is defined as

$$\mathbf{cost}_{\mathbf{ALCFRA}}(e) = MIN + (MAX - MIN)\mathbf{adapt}(e) \tag{5.3}$$

So the parameter $a_e$ modulates the link cost function, making it more convex when the link load is low (so as to balance the load) and making it concave when the link load is heavy (as in the examples in Figure 5.3).

The shape of the link cost function for links with high utilisation is close to the shape of the link cost used in shortest path computation (it uses a constant value of 1). So, if the value of $a_e$ is positive and close to $A$ it means that link utilisation is low and the LD approach is preferred (by the use of an exponential link cost function), but if the value of $a_e$ is negative and close to $-A$ the shape of the link cost function resembles the constant values used in min-hop path computation. This is consistent with the conclusion in [98], that RC algorithms result in a better performance for heavy loaded networks, while LD algorithms are beneficial when the load is light.

The adaptation of the link cost function used in ALCFRA produces an almost flat link cost function around the operating point, located at the mean long term link utilisation. This ensures that transient slight changes in the link utilisation will not significantly affect the link cost, and so the likelihood of traffic fluctuations is reduced.

### 5.1.3 ALCFRA Link Cost Sensitivity

The ALCFRA link cost function is modulated according to the long term link utilisation. So we can observe that there are various sensitivities for various long term link utilisations (various values of parameter $a_e$). Using the definition of link cost sensitivity given by the Eq. (4.3), the sensitivity of the ALCFRA cost function for link $e$ can be written as

$$S_{ALCFRA}(e) = \frac{u(e)}{cost_{ALCFRA}(e)} \frac{\partial cost_{ALCFRA}(e)}{\partial u(e)}, \tag{5 4}$$

and finally

$$S_{ALCFRA}(e) = \begin{cases} \frac{u(e) a_e^{u(e)} \ln a_e}{a_e^{u(e)} - 1}, & \text{if } a_e > 1 \\ 1, & \text{if } -1 \le a_e \le 1 \\ \frac{u(e)(|a_e| - 1)}{\left((|a_e|-1)u(e)+1\right) \ln\left((|a_e|-1)u(e)+1\right)}, & \text{if } a_e < -1 \end{cases} \tag{5 5}$$

The ALCFRA link cost function adapts to the long term link utilisation When this is achieved (i e when $a_e = A\left[u^{tr} - u(e)\right]/u^{tr}$) the ALCFRA link cost sensitivity can be modelled as shown in Figure 5 4

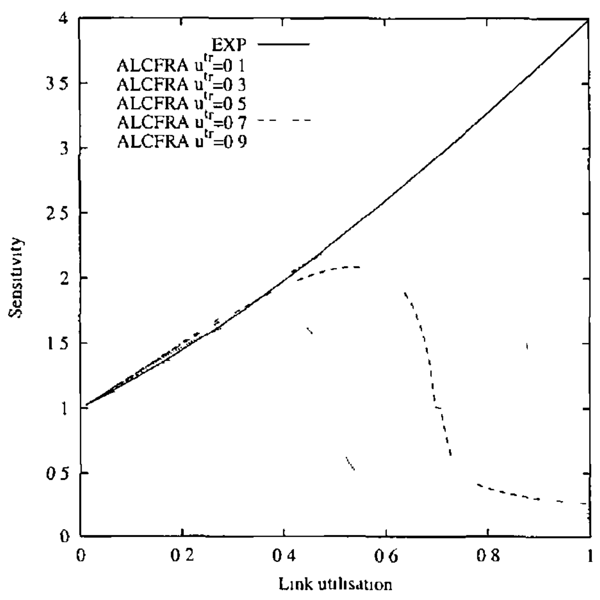


**Figure 5 4** *Link Cost Sensitivity of the EXP and ALCFRA algorithms*

The sensitivity of the link cost function used by the ALCFRA algorithm for highly utilised links is significantly smaller than that of the EXP algorithm So the likelihood of traffic fluctuations is reduced when the utilisation approaches 100% [149]

The sensitivity of the ALCFRA link cost function for stabilised long term link utilisation ($a_e = A\left[u^{tr} - u(e)\right]/u^{tr}$) depends on the parameter $u^{tr}$ – *threshold utilisation* By increasing the value of $u^{tr}$ we can increase the sensitivity of ALCFRA algorithm when required, this is shown in Figure 5 4 So increasing $u^{tr}$ can result in a sensitivity close to that of the LD approach as represented by the EXP algorithm

Using the parameter $u^{tr}$ we can change the level of ALCFRA sensitivity when required However in most cases we set $u^{tr}$ equal to 0 5 since this provides the best tradeoff between stability and load balancing ability

### 5.1.4 ALCFRA first derivative

To check how small changes in link cost affect the link cost value we use the first derivative of the link cost function The first derivative of ALCFRA can be expressed as

$$\frac{\partial cost_{ALCFRA}(e)}{\partial u(e)} = (MAX - MIN)\begin{cases} \frac{\ln a_e}{a_e - 1} {a_e}^{u(e)}, & \text{if } a_e > 1 \\ 1, & \text{if } -1 \le a_e \le 1 \\ \frac{|a_e|-1}{\left((|a_e|-1)u(e)+1\right)\ln|a_e|}, & \text{if } a_e < -1 \end{cases} \qquad (5\ 6)$$

An example of the first derivative of the ALCFRA link cost function (with $MAX = 1$, $MIN = 0$,$A = 100$ and $u^{tr} = 0\ 5$) is shown in Figure 5 5

The link modulation used in ALCFRA effectively reduces the influence of small transient changes of link utilisation on the value of the link cost function, reducing also the likelihood of traffic oscillations

The first derivative of the ALCFRA link cost function, and thus its sensitivity depends on the *threshold utilisation* parameter $u^{tr}$, as shown in Figure 5 6 The smallest increase in the link cost value around the long term link utilisation occurs when we set $u^{tr}$ equal to 0 5 This results in the first derivative of the ALCFRA link cost function passing through the saddle m Figure 5 6 In this case the first derivative is bounded and $\frac{\partial cost_{ALCFRA}(e)}{\partial u(e)} \le 1$

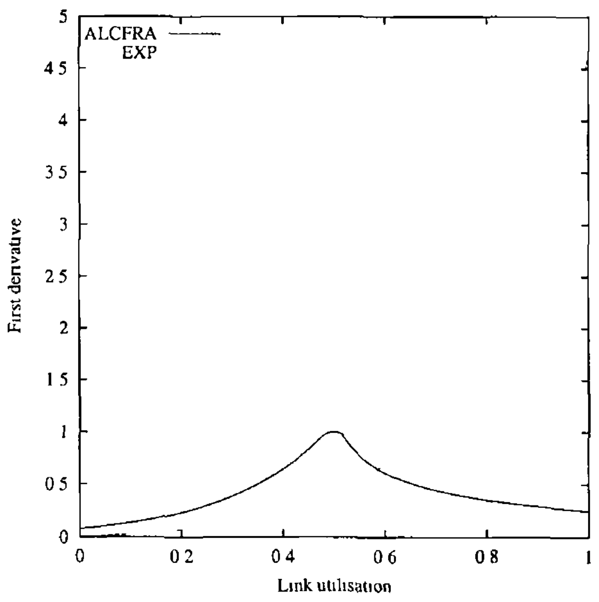


**Figure 5 5** *First derivative of the link cost function for the EXP and ALCFRA algorithms*

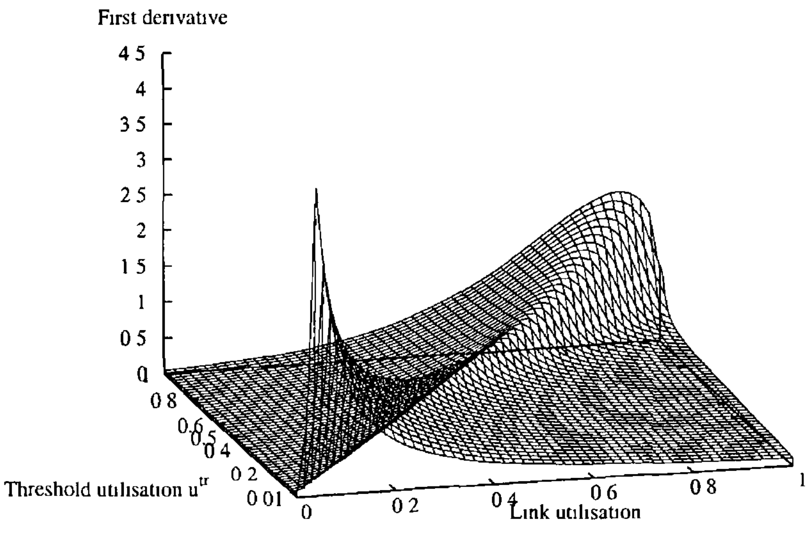


**Figure 5 6** *First derivative of the ALCFRA link cost function for various values of $u^{tr}$*

The observation that $u^{tr}$ should be equal to 0 5 is also confirmed by our simulations, the results of which are described next

## 5 1 5 ACLFRA performance evaluatıon

The LD routıng strategıes usıng a convex lınk cost functıon whıch ıncreases wıth network load such as *shortest-dıstance path* [98] (whıch uses the ınverse avaılable bandwıdth as cost), or algorıthms whıch use an exponentıal lınk cost functıon [19, 65, 82], balance the load well over all lınks Although these strategıes may result ın slıghtly dıfferent performance for dıfferent network topologıes and traffıc models, all of them suffer from traffıc fluctuatıons We compare the stabılıty ımprovement of the LD approach achıeved by applyıng a standard averagıng method and by the ALCFRA algorıthm The LD approach ıs represented by the EXP algorıthm usıng the lınk cost functıon gıven by Eq (3 2) The averaged versıon of the EXP algorıthm we denote as avgEXP

In thıs sectıon we present the benefıts of ALCFRA ın the presence of an ınaccurate envıronment, descrıbıng how the adaptatıon of the lınk cost functıon mechanısm enables the algorıthm to tolerate stale state ınformatıon

### 5 1 5 1 Sımulatıon model

The detaıls of the model used ın our sımulatıons are provıded ın Chapter 3 In thıs sectıon we outlıne those elements of thıs model whıch apply to thıs experıment and the parameters whıch apply specıfıcally to the sımulatıon of ALCFRA

We assume that update polıcıes advertıse a state of the lınk by advertısıng ıts utılısatıon and cost When the EXP algorıthm ıs used new state updates are trıggered ıf the relatıve change ın utılısatıon crosses a predefıned threshold, ı e ıt uses a threshold based update polıcy (see Sectıon 3 4) The ALCFRA algorıthm requıres a dıfferent type of update polıcy to exploıt the advantage of adaptıng the lınk cost functıon accordıng to the long term lınk utılısatıon A new update ıs trıggered ıf the lınk cost value (rather than the utılısatıon) changes sıgnıfıcantly The ALCFRA update polıcy partıtıons the range of lınk cost values ınto classes of equal sıze and every tıme a boundary between classes ıs crossed (sıgnıfyıng that cost has changed) a new link state advertisement is generated In our simulations the link cost was partitioned into 10 equal size classes So the adaptation of link cost function together with the ALCFRA update policy ensures that, when the link utilisation moves slightly from the operating point (long term link utilisation) the link cost remains unchanged However a significant change in link utilisation will affect the link cost, ensuring a fast response to congestion

We evaluate the performance of ALCFRA on the network with the so-called ISP topology [10, 160] shown in Figure 3 3(a) We assume that each link has bandwidth $C$ ($C = 45Mbps$ as in DS-3 links), and the traffic has a uniform pattern, as described in Section 3 3

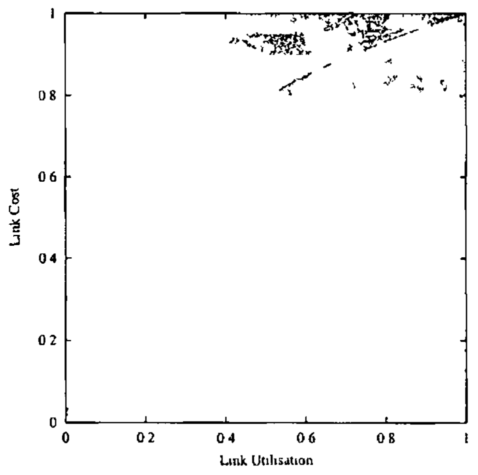
#### 5 1 5 2 ALCFRA - network response

The network response of ALCFRA shown in Figure 5 7 proves that ALCFRA reduces the likelihood of traffic fluctuations We evaluate the network response of one link from the ISP topology for the network load $\rho = 1$, which results in an average link utilisation of 67% when the EXP algorithms is used and of 80% when ALCFRA is used We observe that the area of network response is located close to the operating point when ALCFRA is used However, in the case of the EXP algorithm the area of the network response is situated far from the optimal, which results in unbounded oscillations Moreover, the use of the ALCFRA algorithm results in a higher average utilisation for that link This suggests that reducing fluctuations leads to higher network throughput

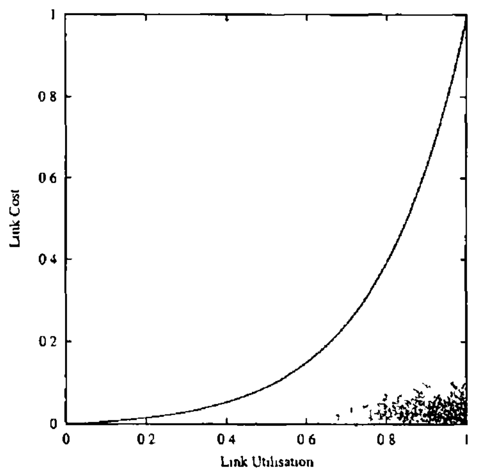
These results apply only for a single link, however they show the general trend in system evolution, whereby the effect of traffic fluctuations is minimised

#### 5 1 5 3 ALCFRA vs EXP vs avgEXP

First we check how the performance of EXP, avgEXP and ALCFRA changes with different levels of network load as recorded in Figures 5 8 and 5 10 In the experiment described below we use the following ALCFRA parameters $A = 100$,



(a) ALCFRA



(b) EXP

**Fıgure 5 7** *Observed network response of the ALCFRA and EXP algorıthms*

$\Delta = 0\ 1$, $u^{tr} = 0\ 5$ (the reason for choosıng these values wıll be gıven later)

When state ınformatıon ıs quıte accurate, hold-down tımers *hd* are set to 1 second, so the avgEXP algorıthm uses an averagıng perıod of 10 seconds to cover more than one lınk state update In thıs case ALCFRA does not perform as well as EXP, whıch achıeves the load balancıng task ın an optımal way, as shown ın Fıgure 5 8 The avgEXP algorıthm does not brıng any ımprovement over EXP (also Fıgure 5 8), because ın thıs case averagıng of lınk cost ıs not requıred – precıse ınformatıon prevents traffıc fluctuatıons However ALCFRA performs almost as good as EXP does, and moreover generates the lowest update overhead, as ıs clear from ın Fıgure 5 9

To ınvestıgate an envıronment when state ınformatıon ıs ınaccurate, the hold-down tımers *hd* are set to 40 seconds (Fıgures 5 10 and 5 11), and the avgEXP algorıthm has to use an averagıng perıod of 80 seconds to cover more than one lınk state update As seen ın Fıgure 5 10 ALCFRA performs as well as or even better than EXP for varıous level of network load The avgEXP algorıthm ıs outperformed by EXP and ALCFRA because of ıts slow response to changes ın the network load The update overhead for all three approaches ıs almost the same (see Fıgure 5 11)

Also, the performance of the ALCFRA algorıthm does not depend strongly

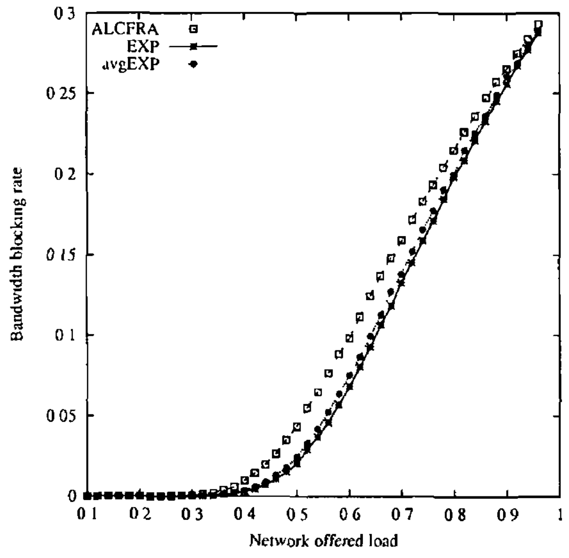


**Figure 5 8** *Call blockmg rate of ALCFRA, EXP and avgEXP versus mcreasmg load, hd* = 1*sec*

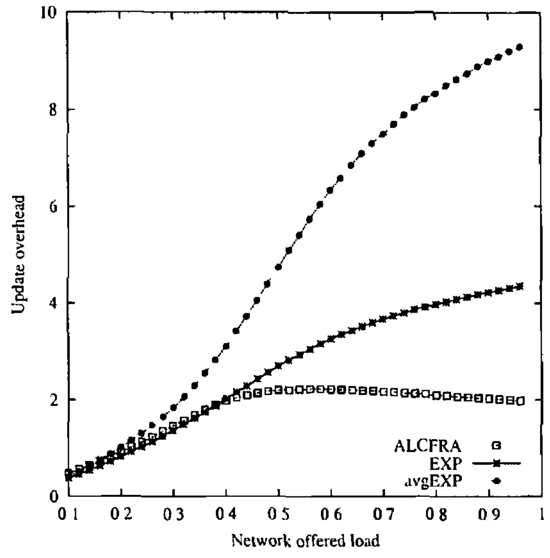


**Figure 5 9** *Update overhead of ALCFRA, EXP and avgEXP versus mcreasmg load, hd* = 1*sec*

on the level of maccuracy (as seen m Figure 5 12 ) The long term lmk utihsation, used to adapt the ALCFRA lmk cost function, is significant, especially when the average connection holdmg time is long (as 180 seconds used m this experiment) This can be used to smooth out traffic fluctuations even if the state information is strongly maccurate

The poor performance of the simple averagmg method shown m Figure 5 12 confirms our suspicions that due to its slow response to changes m lmk state this

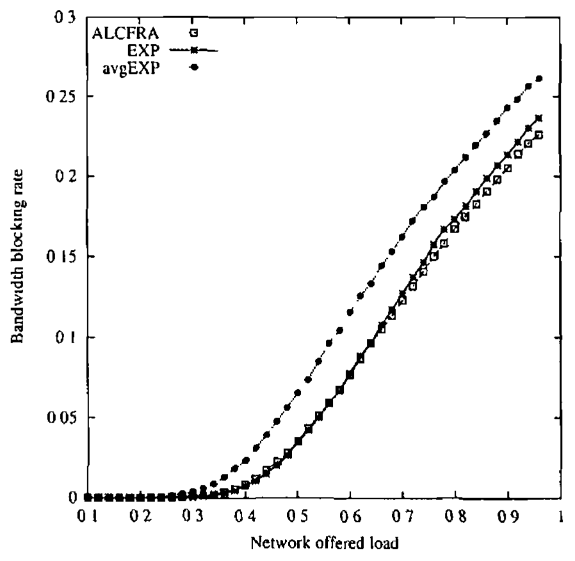


**Figure 5 10** *Call blockmg rate of ALCFRA, EXP and avgEXP versus mcreasmg load, hd =* 40*sec*

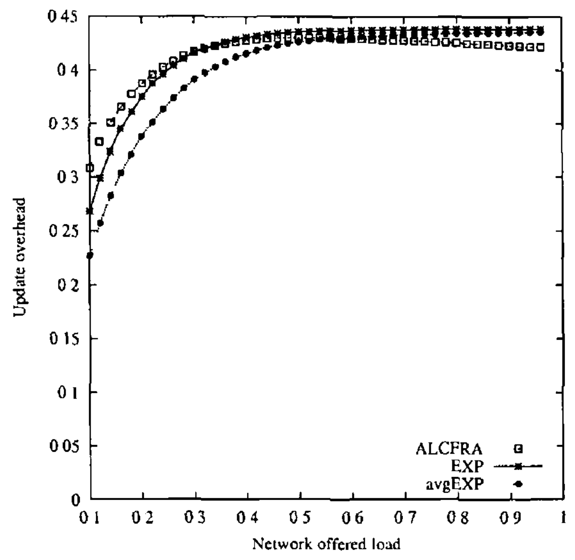


**Figure 5 11** *Update overhead of ALCFRA, EXP and avgEXP versus mcreasmg load, hd =* 40*sec*

method is more suitable for datagram networks than for virtual circuit networks

### 5 1 5 4 The mfluence of traffic on ALCFRA

The performance of the ALCFRA algorithm depends upon the nature of network traffic ALCFRA modulates the shape of the lmk cost function accordmg to the long-term lmk utilisation One of the mfluences of the latter is the holdmg time of

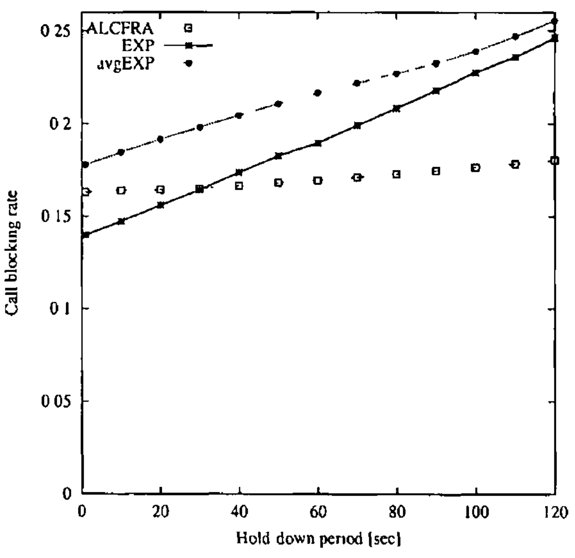


**Figure 5 12** *Call blockmg rate of ALCFRA, EXP and avgEXP versus mcreasmg holdmg penod hd, network load* $\rho = 0\ 8$

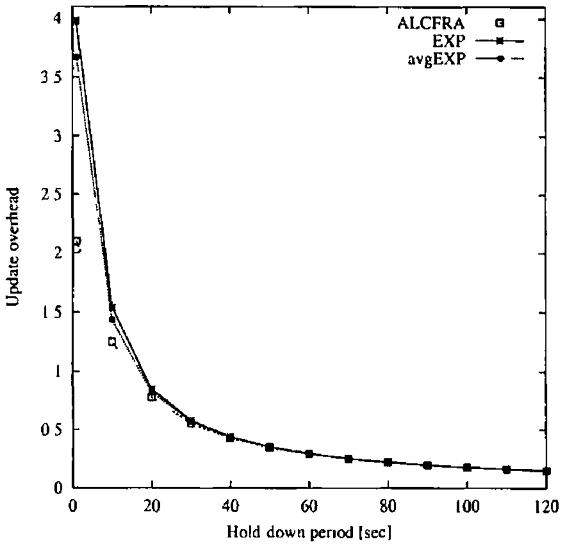


**Figure 5 13** *Update overhead of ALCFRA, EXP and avgEXP versus mcreasmg holdmg penod hd, network load* $\rho = 0\ 8$

connections If the connection holdmg tme is short, the long-term utlisation provides less useful mformation than when connections are of long duration However ALCFRA outperforms pure LD algonthms in the presence of maccuracy, as shown in Figure 5 14, even when connections are of short duration, although its advantage over pure LD algonthms is less pronounced than when connections have longer holdmg times

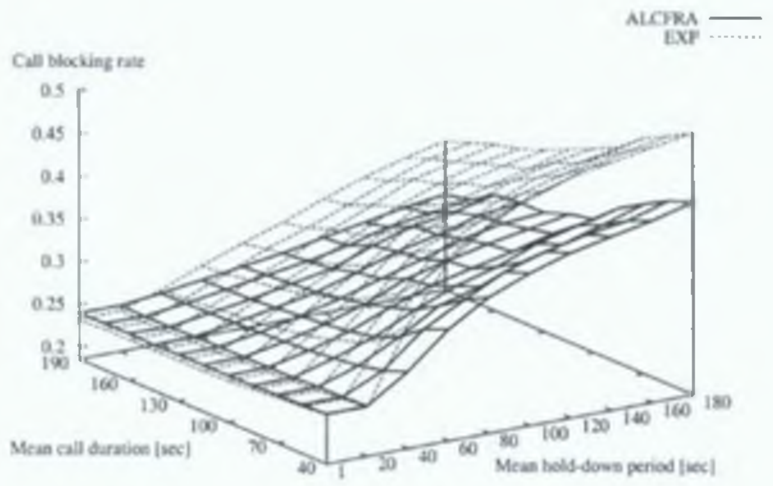


**Figure 5.14:** *Call blocking rate of ALCFRA versus increasing call duration and hold-down timer values*

When the connection durations are long (such as when short lived flows are filtered out as proposed in [132, 134]), ALCFRA results in a very stable performance, and eliminates the oscillation effect almost completely. This can be observed in Figure 5.14 for values of mean call duration above 160 sec. In this case the ALCFRA algorithm, unlike the standard EXP algorithm, offers constant performance, which does not depend on the level of inaccuracy.

### 5.1.6 The tuning of ALCFRA

The modulation of the link cost function performed by ALCFRA can be adjusted using three parameters, namely: $A$, $\Delta$ and $u^{tr}$.

#### 5.1.6.1 The parameter $A$

$A$ is a large positive constant (see [19, 65]) although its value cannot be too large, because this would result in a too rapid change of the link cost when the utilisation approaches the total bandwidth. Also its value cannot be too small, because this would cause traffic to be redirected from paths with fewer numbers of hops to longer paths following slight increase in utilisation We have set $A$ equal to 100 in our simulations A more detailed discussion about the choice of $A$ is given in [65]

#### 5 1 6 2 The parameter Δ

The choice of a value for the parameter Δ determines how sensitive the adaptation of the ALCFRA link cost function is to changing traffic conditions If the value of Δ is large, the shape of the ALCFRA link cost function is dictated largely by the recent history of link utilisation rather than the long-term trend If the value of Δ is small, the shape of the ALCFRA link cost function corresponds to the long-term link utilisation This suggests that a small value of Δ should be used, although the exact number which gives the best performance must be determined by simulation The period of observation of the long-term link utilisation used by ALCFRA is inversely proportional to Δ As shown in Figure 5 15, longer periods of observation of link utilisation (smaller values of Δ) result in the smallest call blocking rate Therefore the optimal values are very small We set Δ equal to 0 1 for all our simulations

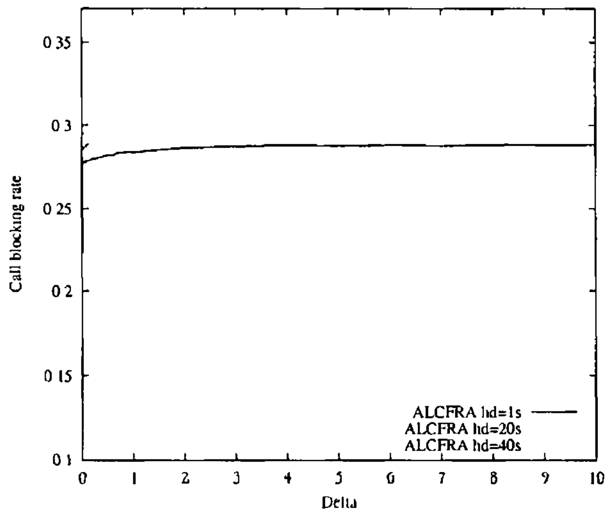


**Figure 5 15** *Call blocking rate of ALCFRA versus increasing Δ*

### 5 1 6 3 The parameter $u^{tr}$

The choıce of the value for the threshold utılısatıon $u^{tr}$ determınes what ıs the balance between the use of convex and concave lınk cost functıons To be more precıse, the value of $u^{tr}$ sets the poınt for whıch the shape of the lınk cost functıon ıs swıtched from convex to concave accordıng to the long-term lınk utılısatıon

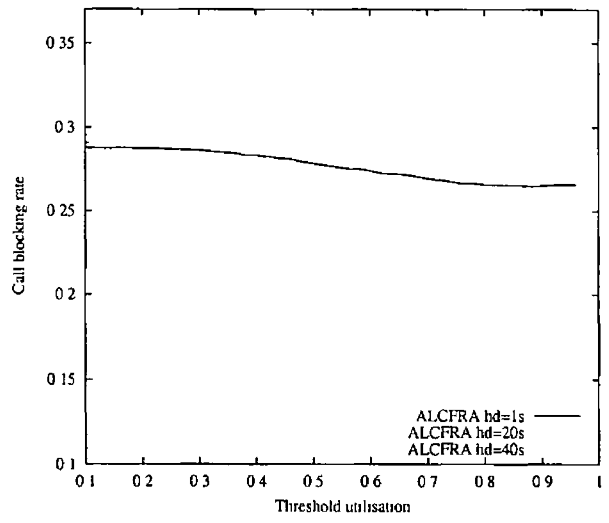


**Fıgure 5 16** *Call blockıng rate of ALCFRA versus ıncreasıng threshold utılısatıon*

When the state ınformatıon ıs precıse and we ıncrease the threshold probabılıty $u^{tr}$ (see Fıgure 5 16) we observe that the blockıng rate decreases Thıs suggests that when workıng wıth accurate network state ınformatıon we can benefıt from load balancıng approaches (approaches usıng an exponentıal lınk cost functıon)

When state ınformatıon ıs ınaccurate (ı e when longer hold-down perıods are used) we observe (see Fıgure 5 16) that the blockıng rate grows rapıdly when $u^{tr} > 0\ 5$ Thıs growth ıs caused by traffıc fluctuatıons whıch are lıkely to occur for algorıthms usıng an exponentıal lınk cost functıon when the network load ıs hıgh and the state ınformatıon ıs ınaccurate Therefore we need to set $u^{tr}$ to 0 5, to swıtch to a convex lınk cost functıon when such oscıllatıons are expected Moreover, as seen ın Fıgure 5 5, choosıng a value of 0 5 for $u^{tr}$ results ın the smallest ıncrement of the value of the ALCFRA lınk cost functıon around long term lınk utilisation

### 5 1.7 ALCFRA summary

The ALCFRA algorithm is more complex than the standard LD algorithms It introduces requirements to keep track of the long-term link utilisation and to modulate the shape of the link cost function in response to this However this complexity is justified by the reduction of traffic fluctuations, which cause the performance of LD algorithms to deteriorate

Overall, ALCFRA can be used in networks where the LD approach is preferred and state information is imprecise Although it results in a higher than optimal blocking rate when state information is up-to-date, it provides a stable path selection mechanism which is tolerant of inaccuracies in state information

## 5.2 Connectivity Aware Routing (CAR)

In Section 4 2 we have observed that in networks with low connectivity, the LD approach performs better, because it uses scarce resources in a more efficient way than the RC approach However, for highly connected networks, which usually also feature many min-hop paths, the RC approach performs better because it reduces the likelihood of traffic fluctuations

Our goal is to design an algorithm which would benefit from both approaches and provide excellent performance for a wide range of network topologies This Connectivity Aware Routing (CAR) algorithm provides a tradeoff between the extremes of the LD and RC approaches

When a router using the LD approach chooses a path, it does so in a greedy fashion and takes the shortest path available without regard to how its choice will affect other users When traffic is light, this approach works fine However, when traffic levels increase, the routes selected in a greedy fashion are likely to compete for the available resources Under heavy loads, the goal of routing should be to give the average route a good path instead of attempting to give all routes the best path. Some of the routes should use longer paths so that remaining routes can make effective use of overloaded links. The routes using longer paths should be those that have alternate paths which are only slightly longer. However the length of alternative paths depends strongly on the network topology.

The CAR algorithm tries to exploit the dependence of LD and RC performance on the level of network connectivity. So it modifies the load balancing ability according to the level of network connectivity, by increasing the *bias* factor described in Section 2.2.4.1. The CAR algorithm is also motivated by the observation that stability is less crucial in networks with lower connectivity (see Section 4.2). When there is only one path between any source and destination, traffic cannot be switched to an alternative path. However, if there are many such paths traffic oscillations are more likely to occur. Hence the CAR algorithm increases the value of *bias* with the level of network connectivity.

### 5.2.1 CAR details

The performance of the LD approach suffers a lot from traffic fluctuations. Such oscillations can be damped by adding a positive constant, called *bias*, to the link cost function (see Section 2.2.4.1). It was observed that by adding a large *bias* the routing becomes fairly stable, but it also loses the ability to avoid congestion [29].

We think that the value of *bias* should increase with the level of network connectivity, because LD methods are beneficial for networks with low connectivity, and for highly connected networks they are outperformed by RC algorithms.

Using the assumptions from Section 3.1.2, we also assume that the routing algorithm performs load distribution using a normalised exponential link cost function given by Eq. (3.2). In our hybrid approach (CAR), we add a constant $\alpha$ called *bias* to this cost function, so that the primary link cost metric of link $e$ is expressed by

$$\text{cost}_{\text{CAR}}(e) = (MAX - MIN)(\alpha + (1 - \alpha)\exp(e)) \qquad (5\ 7)$$

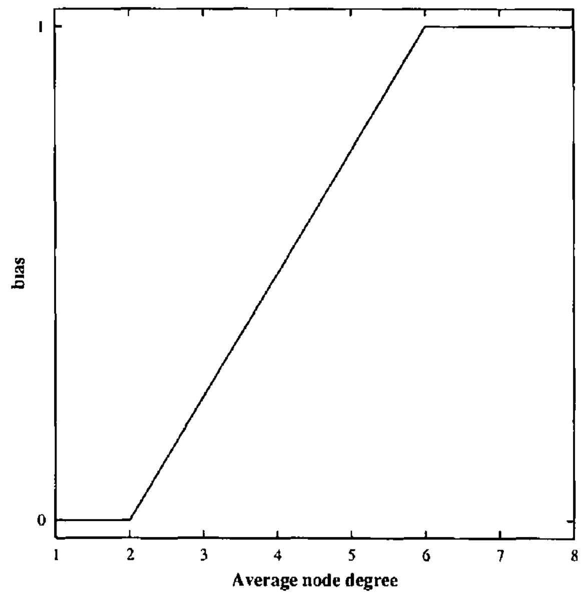
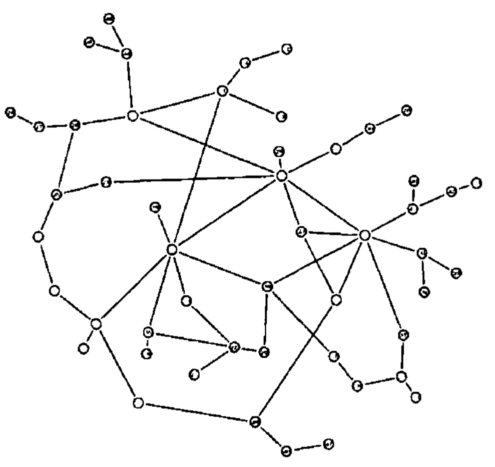
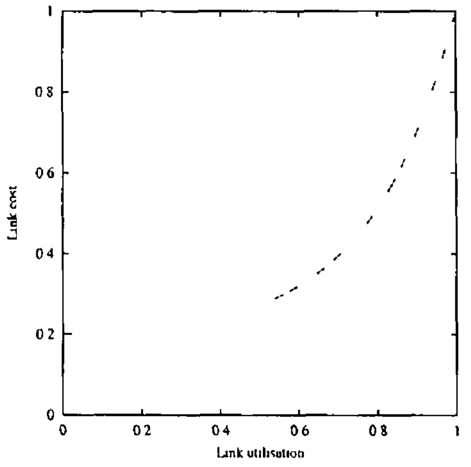


**Figure 5 17** *Values of bias $\alpha$ for the CAR algorithm*

Two examples of lmk cost functions, for which the value of *bias* was set by the CAR algorithm accordmg to the connectivity level of the correspondmg topology, are shown in Figures 5 18 and 5 19

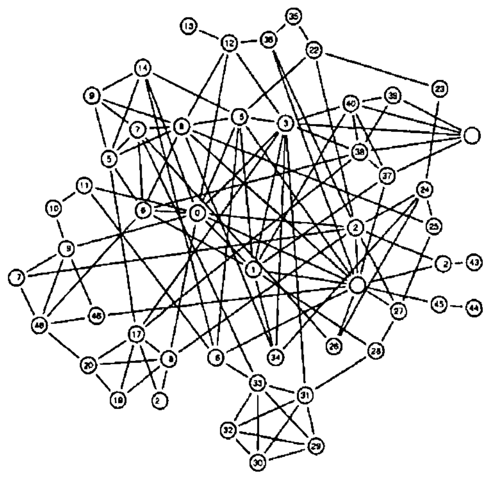
As we explain above, the value of bias $\alpha$ should increase with the level of connectivity We decided to vary $\alpha$ as a function of average node degree in the manner shown in Figure 5 17 This function has been chosen to yield the LD approach for an average node degree $< 2$, and the RC approach for an average node degree $> 6$ In Section 4 2 we have shown that the respective approaches give the best performance in these two situations We assume that the task of setting the bias $\alpha$ according to the connectivity of the network is performed by the administrator Initially this would be done as per Figure 5 17 However, the administrator can change $\alpha$ if he decides that more or less load balancing is required Since the level of network connectivity will not change very often, this task will need to be done only ınfrequently

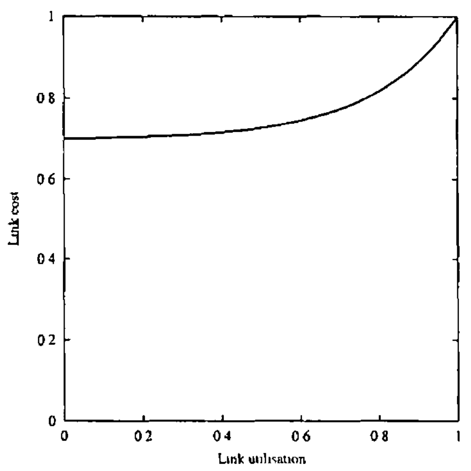

(a) Network



(b) Cost functıon

**Fıgure 5 18** *A low connectıvıty network and the correspondıng CAR cost functıon*


(a) Network



(b) Cost functıon

**Fıgure 5 19** *A hıgh connectıvıty network and the correspondıng CAR cost functıon*

For each lınk the prımary cost ıs calculated as descrıbed above However, the computatıon of the least cost path can result ın a set contaınıng more than one path of equal cost The second lınk cost metrıc ıs used to spread the traffıc between them For sımplıcıty we decıded that the path wıth the hıghest resıdual bandwıdth should be chosen from the set of least cost paths

The CAR algorıthm may be summarısed as follows For each lınk the value of resıdual bandwıdth ıs advertısed usıng a floodıng mechanısm The frequency of such advertısements ıs controlled by the lınk state update polıcy [10] (ın our experıment we use a threshold based polıcy controlled by a *hold-down* tımer) Based on these advertısements each router calculates the cost of all lınks usıng Eq (5 7) Upon receıvıng a route request the set of least cost paths ıs computed If there ıs more than one such path, the one wıth the hıghest resıdual bandwıdth ıs chosen If the reservatıon along a chosen path can be realısed, the request ıs accepted, and otherwıse ıt ıs rejected

## 5 2.2 CAR Link Cost Sensitıvıty

The lınk cost functıon used by the CAR algorıthm ıs the sum of the bıas and the normalısed exponentıal lınk cost functıon gıven by Eq (3 2) So the sensıtıvıty of the cost functıon of lınk $e$ when the CAR algorıthm ıs used can be wrıtten as

$$S_{CAR}(e) = \frac{u(e)}{cost_{CAR}(e)} \frac{\partial cost_{CAR}(e)}{\partial u(e)} = \frac{(1-\alpha)u(e)A^{u(e)} \ln A}{\alpha(A-1) + (1-\alpha)(A^{u(e)}-1)} \qquad (5\ 8)$$

Examples of lınk cost sensıtıvıty of the CAR algorıthm (for the case where $MAX = 1$, $MIN = 0$ and $A = 100$) are shown ın Fıgure 5 20 We observe that when the CAR algorıthm uses a large value of $\alpha$, the lınk cost sensıtıvıty ıs sıgnıfıcantly reduced Hence ıt responds to changıng network loads only ın a restrıcted manner Even quıte small values of $\alpha$ decrease lınk cost sensıtıvıty sıgnıficantly compared wıth the EXP algorıthm, as shown ın Fıgure 5 20 Therefore large values of $\alpha$ are used by the CAR algorıthm only when network connectıvıty ıs rıch, allowıng the load to be effectıvely balanced over the set of mın-hop paths

When we compare the fırst derıvatıve of the CAR and EXP cost functıons (as ın Fıgure 5 21) we can see that the CAR cost functıon grows much more slowly than the EXP cost functıon because of the bıas factor So transıent slıght changes ın the lınk utılısatıon affect the lınk cost value much less than for the EXP algorıthm

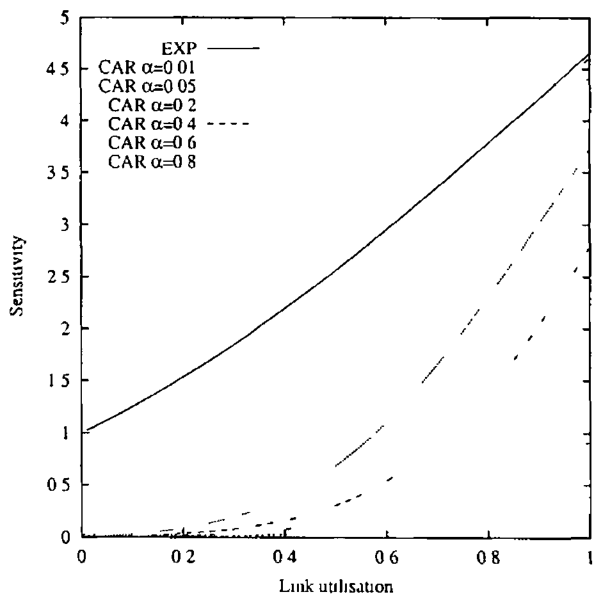


**Fıgure 5 20** *Lınk Cost Sensıtıvıty of the EXP and CAR algorıthms*

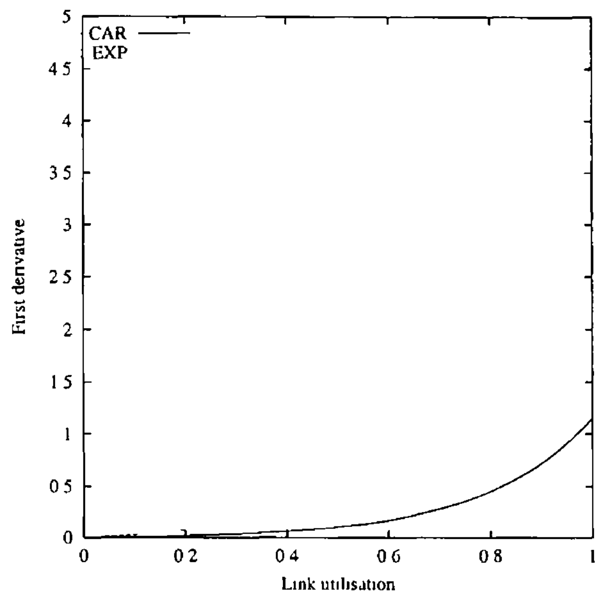


**Fıgure 5 21** *Fırst derıvatıves of the CAR and EXP lınk cost functıons*

## 5 2.3 CAR performance evaluation

We compare the performance of the *connectıvıty aware routıng* (CAR) algorıthm wıth a representatıve of the RC approach – the WSP algorıthm, and wıth the EXP algorıthm as a representatıve of the LD approach

### 5 2 3 1 Simulation model

The details of the model used in our simulations are provided in Chapter 3 However, in this section we outline the specific elements of this model which apply to this experiment or these which differ from the original The simulation environment used in this experiment extends the one presented in Section 4 2 1 1

The goal of our experiment is to show how the CAR algorithm improves on the LD and RC approaches represented respectively by the EXP and WSP algorithms Our experiment covers a wide range of network connectivities and levels of inaccuracy of state information Thus we require a network topology whose connectivity can be adjusted in a controlled manner To achieve this, we decided to use a base network shown in Figure 3 3(b) and we extended in as described in Appendix 3 2 and Section 4 2 1 1

To verify that our performance results do not reflect some peculiarity in this methods for synthesising a network topology, we also simulate two randomly generated networks each with a different level of connectivity which were created using the GT-ITM package [161] Both networks have 50 nodes The first one, shown in Figure 3 3(c), has an average node degree of 2 3, while the second network topology, shown in Figure 3 3(d), has an average node degree of 4 96

In all the networks we use bidirectional links, each with identical capacity $C$ (in our simulations we use DS-3 links with capacity $C = 45Mbps$) We use two traffic patterns, uniform and bursty, as presented in Section 3 3

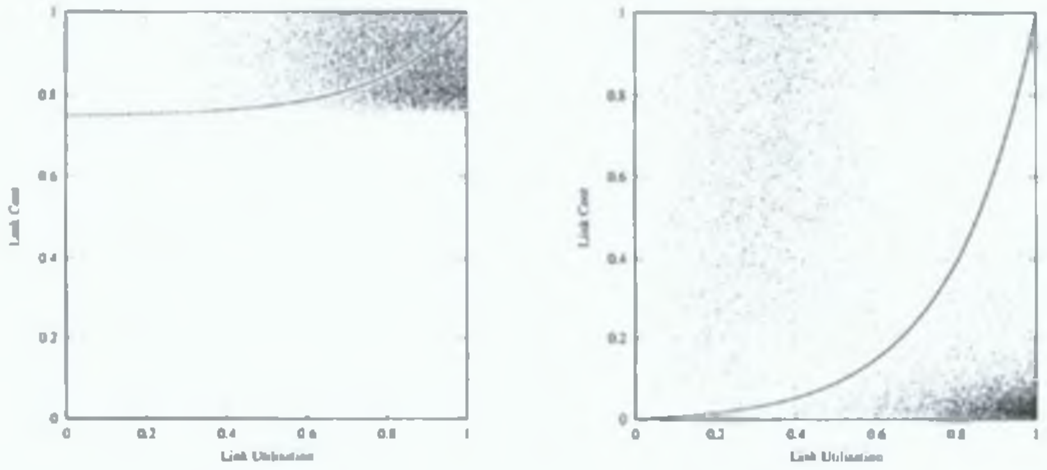
Apart from different connectivity levels we also model inaccuracy in state information through the use of *hold-down* timers which set the minimal time interval between concurrent link state updates (we denote this interval as *hd*) We assume that each node uses a threshold based update policy [10, 160] with the threshold trigger of 10% controlled by a hold-down timer We use only one hold-down timer for all links outgoing from the node

#### 5.2.3.2 CAR – network response

The network response of CAR shown in Figure 5.22 illustrates that CAR reduces the frequency of traffic fluctuations. We evaluate the network response of one link from the ISP topology for the network load $\rho = 1$, which results in an average link utilisation of 67% when the EXP algorithm is used and of 82% when CAR is used. For the CAR algorithm we use $\alpha = 0.75$.

We observe that the area of network response is located close to the operating point (an average link utilisation) when CAR is used. We also see that the area of network response is bounded due to the use of the bias factor. This bounds traffic oscillations and improves stability. When the EXP algorithm is employed the area of network response is situated far from a cost function, which results in unbounded fluctuations.

Moreover, the use of CAR algorithm results in higher average utilisation for the observed link. This suggests that reducing fluctuations leads to higher network throughput.



(a) CAR (b) EXP

**Figure 5.22:** *Observed network response of the CAR and EXP algorithms*

In this section we presented the network response of a single link. This gives a general idea about the reduction of traffic fluctuations achieved by the CAR algorithm. However, to draw general conclusions we need to analyse the performance of the whole network – not just a sıngle lınk This ıs done ın the followıng sectıon

### 5 2 3 3 CAR performance

Here we demonstrate that by settıng an approprıate ratıo between the RC and LD approaches accordıng to the level of network connectıvıty, we can perform stable routıng and maıntaın a low blockıng rate usıng CAR algorıthm

The use of resources under bursty traffıc ıs ıllustrated ın Fıgure 5 23, whıch shows the *relatıve average path length* as a functıon of connectıvıty The EXP algorıthm, by tryıng to avoıd congested lınks, uses paths wıth a non-mınımal number of hops The path length of the WSP algorıthm asymptotıcally approaches the value of 1, whıch confırms that ıt mınımıses resource consumptıon The hybrıd approach used ın the CAR algorıthm selects non mın-hop paths when the level of connectıvıty ıs low, whıle reducıng the use of excessıve resources when connectıvıty ıs hıgh All three algorıthms start wıth the value of *relatıve average path length* below 1, because for networks wıth very low connectıvıty short paths are more lıkely to be successfully establıshed, than longer ones

In Fıgure 5 24 we show the varıatıon ın lınk utılısatıon, whıch ıllustrates how well the network traffıc ıs balanced over all lınks (the smaller the varıatıon, the better load balancıng ıs performed) The EXP algorıthm ıs effectıve at balancıng the load, but due to ıts use of non mın-hop paths ıt gıves rıse to a hıgh blockıng rate (Fıgure 5 28) The WSP performs load dıstrıbutıon poorly, sınce ıt ıs restrıcted to the use of shortest paths The CAR algorıthm provıdes a compromıse between these two extremes, reducıng ıts load balancıng abılıty for hıghly connected networks to keep the blockıng rate low (see Fıgure 5 28)

So as seen ın Fıgures 5 23 and 5 24 the CAR algorıthm reduces ıts load balancıng abılıty and resource consumptıon when the level of connectıvıty ıncreases Thıs results ın an ıntermedıate performance when the network load or the level of ınaccuracy are low (see Fıgures 5 25, 5 26 and 5 27) However ın such sıtuatıons

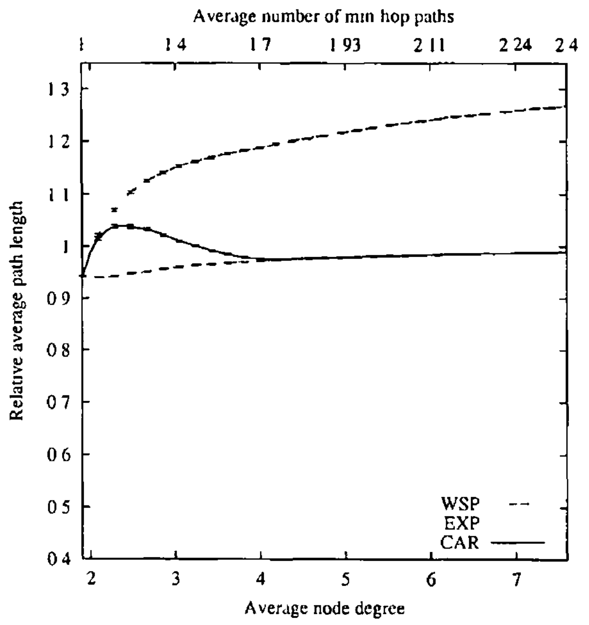


**Figure 5 23** *Relative average path length of WSP, EXP and CAR versus increasing connectivity,* $\rho = 0\ 8$, $hd = 40sec$

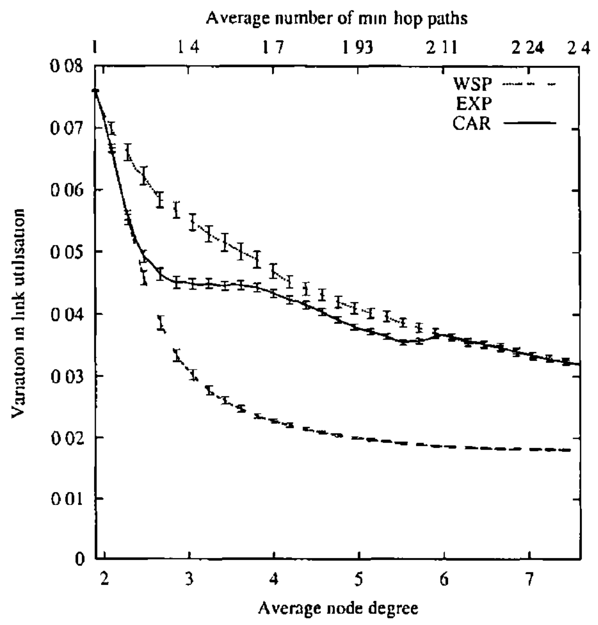


**Figure 5 24** *Variation in link utilisation of WSP, EXP and CAR versus increasing connectivity,* $\rho = 0\ 8$, $hd = 40sec$

traffic fluctuations occur infrequently The advantage of using the CAR algorithm becomes apparent when the network load is high and state information is inaccurate, as shown in Figure 5 28 Due to its use of the *bias* factor the CAR algorithm does not suffer so much from the effects of imprecise state information and offers excellent performance for a wide range of network topologies

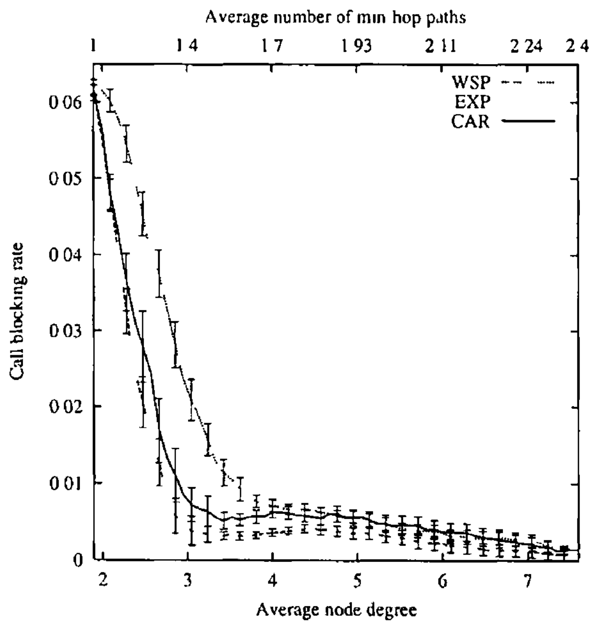


**Figure 5 25** *Call blockmg rate of WSP, EXP and CAR versus increasing connectivity, network load* $\rho = 0\ 3$, $hd = 1sec$

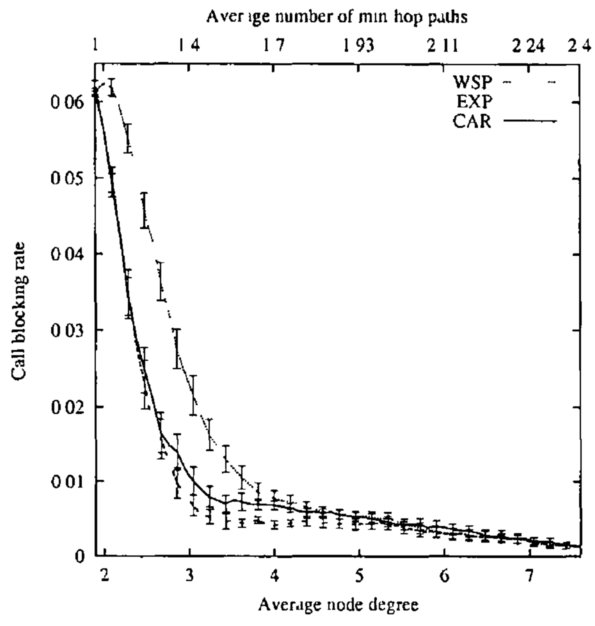


**Figure 5 26** *Blocking rate of WSP, EXP and CAR versus increasing connectivity, network load* $\rho = 0\ 3$, $hd = 40sec$

The WSP algorithm produces good results for highly connected networks and is robust to imprecise state information (Figures 5 26 and 5 28) However, if the level of connectivity is low (when there is usually only one min-hop path between any two nodes) WSP cannot utilise resources as well as either EXP or CAR

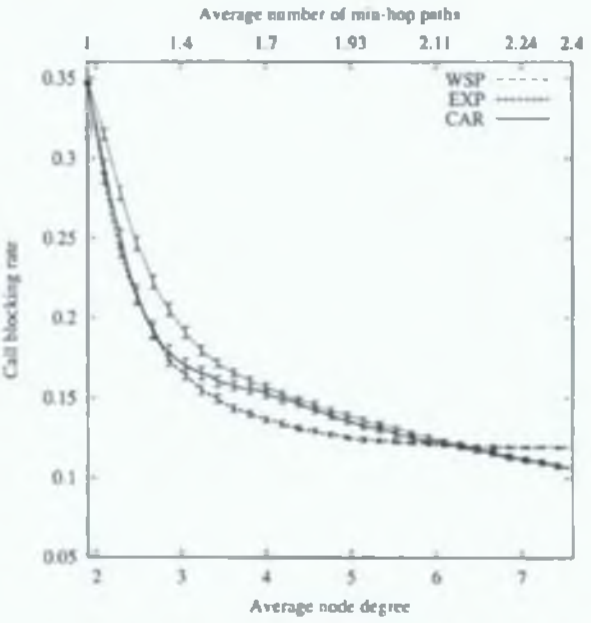


**Figure 5.27:** *Blocking rate of WSP, EXP and CAR versus increasing connectivity; network load* $\rho = 0.8$, $hd = 1sec$

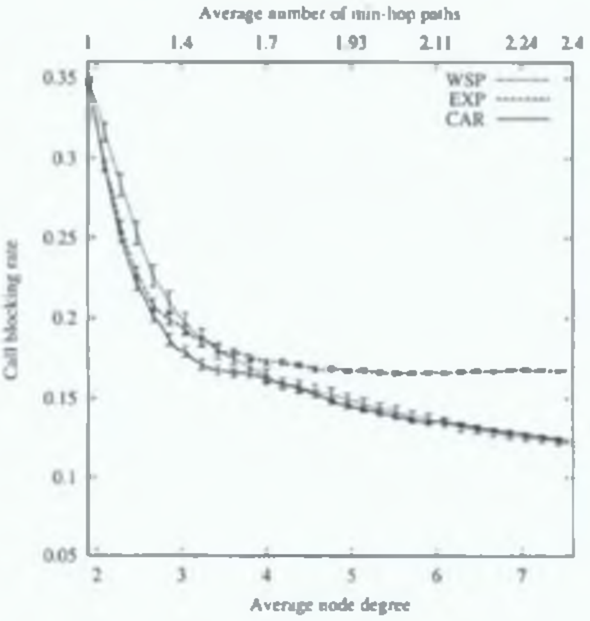


**Figure 5.28:** *Blocking rate of WSP, EXP and CAR versus increasing connectivity; network load* $\rho = 0.8$, $hd = 40sec$

These results show that the CAR algorithm, by adapting its link cost metric according to the level of network connectivity, provides a good routing strategy over a wide range of network topologies. Also, by adding a *bias* proportional to the connectivity level it improves the routing stability, when this is required.

#### 5.2.3.4 Case of network with low connectivity

In this section we compare the call blocking probability of CAR, EXP and WSP for an example network with low connectivity, shown in Figure 3.3(c).

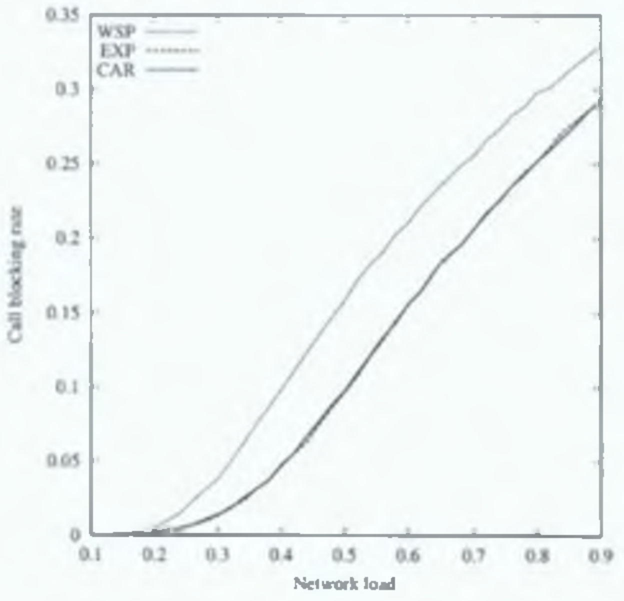


**Figure 5.29:** *Blocking rate of WSP, EXP and CAR versus increasing load for a network with low connectivity, hd = 1sec*

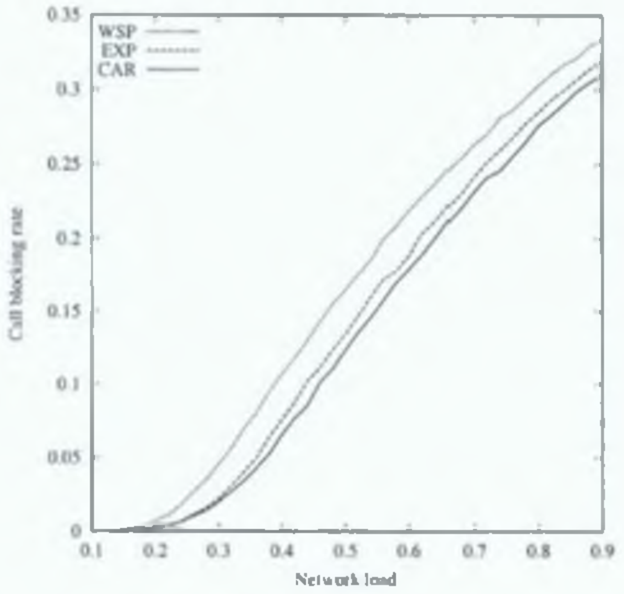


**Figure 5.30:** *Blocking rate of WSP, EXP and CAR versus increasing load for a network with low connectivity, hd = 80sec*

In Figure 5.29 we observe that when state information is up-to-date, CAR and EXP perform load balancing equally well for a wide range of network loads. The

WSP algorıthm, due to ıts use of only the set of mın-hop paths, cannot fully utılıse resources, so ıt gıves rıse to a hıgher blockıng rate

If state ınformatıon ıs ımprecıse, as ın Fıgure 5 30 when $hd = 80sec$, the CAR algorıthm outperforms EXP due to ıts use of the *bıas* factor The EXP algorıthm stıll gıves a lower blockıng rate than WSP, whıch confırms our fındıngs descrıbed ın Sectıon 4 2, that for networks wıth low connectıvıty, the LD approach outperforms RC

#### 5 2 3 5 Case of network wıth hıgh connectıvıty

In thıs sectıon we compare the call blockıng probabılıty of CAR, EXP and WSP for an example network wıth hıgh connectıvıty, shown ın Fıgure 3 3(d)

When routıng algorıthms operate on hıghly connected networks, our fındıngs from Sectıon 4 2 suggest that RC should be preferred Indeed as shown ın Fıgures 5 31 and 5 32, WSP results ın a lower blockıng rate than EXP over a wıde range of loads The EXP algorıthm performs better than WSP only when state ınformatıon ıs precıse and the load ıs low (e g when loads are lower than 0 5 ın Fıgure 5 32) but the advantage ıs slıght

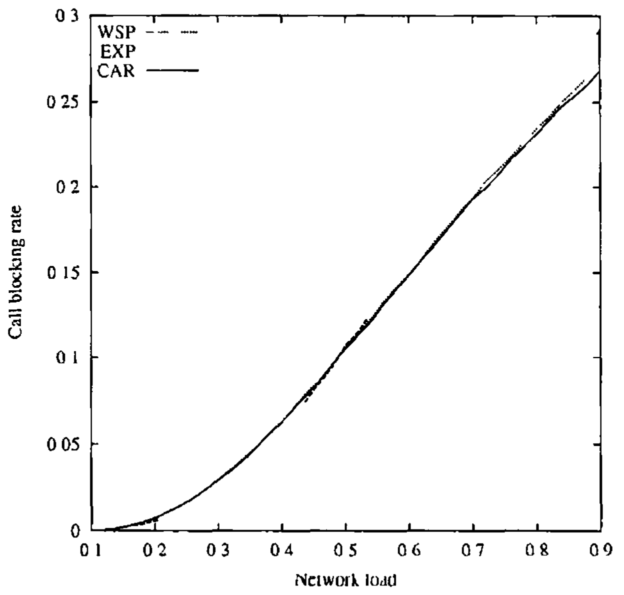


**Fıgure 5 31** *Blockıng rate of WSP, EXP and CAR versus ıncreasıng load for a network wıth hıgh connectıvıty, $hd = 1sec$*

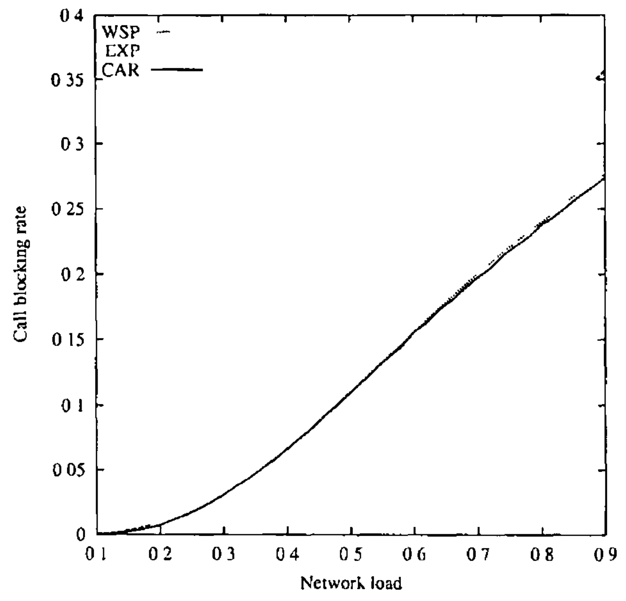


**Fıgure 5 32** *Blockıng rate of WSP, EXP and CAR versus ıncreasıng load for a network wıth hıgh connectıvıty, hd* $= 80sec$

The performance of the CAR algorıthm for hıghly connected networks ıs almost the same as WSP Thıs ıs the result of usıng a large *bıas* factor, whıch lımıts the ınfluence of the load dıstrıbutıng mechanısm Thıs ıs benefıcıal for networks wıth a hıgh connectıvıty level

The CAR algorıthm consıders only the level of network connectıvıty, assumıng that ıt ıs proportıonal to the number of mın-hop paths However, regular topologıes such as the cube [133] feature many mın-hop paths between any source and destınatıon, so the proportıon between connectıvıty and number of mın-hop paths ıs dıfferent than observed here Such topologıes occur ınfrequently ın real networks, but ın such case our observatıons suggest that the load balancıng abılıty of the CAR algorıthm should be further restrıcted

## 5.3 Summary

Traffıc oscıllatıons are often observed ın networks when adaptıve routıng ıs used Thıs leads to network ınstabılıty, lımıts the throughput and consumes excessıve transmıssıon and computatıonal resources The methods for dampıng such fluctuations proposed in [29] and [85] for datagram networks cannot be directly applied to virtual circuit networks Therefore we have presented two methods to improve the stability of link state QoS routing which are suitable for use in such networks

The first method is called Adaptive Link Cost Function Routing Algorithm (ALCFRA) It is designed to improve the stability of load distributing (LD) algorithms in sparsely connected networks, when such algorithms are of the most benefit ALCFRA provides a novel method of incorporating long term link utilisation into the link cost metric Unlike standard averaging methods it does not slow down the speed of response to congestion, and allows the network performance to be stabilised even under highly varying traffic conditions

The second method, called Connectivity Aware Routing (CAR), is suitable for networks of a general topology It improves the stability and reduces excessive consumption of resources by the use of a bias factor adjusted according to the connectivity level of the network This offers significant improvements in network performance For sparsely connected networks it allows good load balancing and the effective utilisation of resources even when the network topology is irregular For highly connected networks it forbids the usage of long paths, since in most cases the load can be balanced over the set of min-hop paths This conserves resources for future connections and stabilises the path selection process

The routing methods presented in this chapter give an important insight into the process of controlling the stability of link state QoS routing methods Our simulations show that by appropriate usage of these methods we can effectively improve the stability and achieve good routing performance for a wide range of network topologies

# CHAPTER 6

# Problems in the deployment of QoS routing

Even should a single QoS algorithm be standardised for use in future multi-service data networks, there would still be various problems with the worldwide deployment of QoS routing We have described some of these in Section 2 2 5, including the growing size of routing tables, the granularity of the path selection process, the efficient aggregation of the network topology for one or more QoS metrics, and the implementation of other mechanisms supporting QoS

In this chapter we focus on two other problems which can be encountered during the deployment of QoS routing algorithms

**Problem 1**

Topology aggregation has been proposed as a method which allows scalability to large networks, and several aggregation methods have proposed by researchers When implementing aggregation we usually have a choice of methods according to the performance goals which we want to achieve (if we want to get accurate path selection, we should use a detailed representation, if we want to reduce communication overhead, we should use a coarser representation) For example, PNNI does not restrict aggregation to only one technique, stating only that "the symmetric star topology is used as the 'default node representation'" [12] Therefore we expect that various various topology aggregation methods can coexist in a hierarchical network However such a free hand given to peer group administrators allows them to hide details of their network topology, degrading the route selection process of other peer groups Therefore in Section 6 1 we propose an efficient method which regulates the way in which such peer groups should cooperate

**Problem 2**

We cannot expect the future QoS-enabled Internet to provide homogeneous QoS support Instead it will be composed of islands which feature QoS mechanisms and islands unaware of QoS This makes it hard to reserve resources along the path selected by the QoS routing algorithm, because some path segments, referred to here as *reservation gaps,* may not support reservation mechanisms Therefore in Section 6 2 we propose two algorithms to improve the reliability of the path selection process which are aware of the QoS support provided by nodes

Both problems arise due to the heterogeneity of the Internet, a property which is unlikely to disappear as the Internet evolves Even if we focus on a single point of time, the Internet remains highly heterogeneous This heterogeneity is considered as "a key property that makes it difficult to model and simulate the Internet" [56]

## 6.1 Coexistence of various topology aggregation methods

Topology aggregation is used in hierarchical networks It maps the detailed topological information about group of nodes into a compact representation Therefore it reduces not only the amount of state information but also the amount of memory and time required to compute paths So topology aggregation is considered a key concept to achieve scalability of QoS aware models in the Internet [153] Topology aggregation methods associated with PNNI hierarchial routing have been described in Section 2 1 4

Both update policies and topology aggregation result in inaccurate state information being maintained by the routers [73], which usually has a negative impact on the path selection process [133] However it has been shown [75] that topology aggregation can have positive influences and can increase the stability of route selection by decreasing traffic oscillations

In this section we consider a two level hierarchical network and consider the effects on routing performance of having a variety of aggregation methods coexisting in the network

## 6.1.1 Aggregation methods

A lot of attention has been devoted to the problem of the most accurate representation of an aggregated topology with one or more QoS constraints [88, 96, 158] The most commonly used aggregation methods are these proposed for ATM networks [91] symmetric star, full-mesh and complex node representation The symmetric star provides little information and full-mesh is not scalable, the complex node representation provides a tradeoff between these two methods The complex node representation is usually created by reducmg the number of edges of the corresponding full-mesh topology [91] For example the *t-spanner* [21] is a compact representation which features a minimal difference between the complex node and its original full-mesh representation

In this thesis we consider only one QoS metric – the available bandwidth, since the aggregation of multiple metrics is a difficult task [88, 96] We assume that accurate information about the number of hops is also provided using a full mesh representation [75], since some algorithms also require information about the number of hops. Because the information about the number of hops changes infrequently, such precise advertisements only slightly increase the overhead.

Since PNNI does not restrict aggregation to only one technique, defining only the symmetric star topology as the default node representation [12], we assume that several aggregation methods can coexist in one hierarchical network at the same time. In this thesis we assume that three aggregation models coexist:

**exact representation** when a group of nodes is abstracted using a full mesh representation or another form of compressed full mesh which allows the original full mesh to be recreated;

**fine representation** when only information about average resource availability (mean available bandwidth) is abstracted using a symmetric star representation;

**coarse representation** when a group of nodes is configured not to advertise information about the availability of its resources or sends such advertisements very infrequently.

### 6.1.2 Aggregation strategies

The Internet is administered by many entities which often operate without any coordination and behave in a greedy manner [52]. Game theory is a suitable tool which can be used to model such a competitive environment [52, 64, 114, 126, 136].

The aggregation of network topology information can also be viewed as a competitive game. This is a valid approach in the case of an internetwork which is a coalition of networks administered by separate entities. Each administrator can choose a different aggregation method based on the selected strategy. We assume that administrators of each peer group do not cooperate and select one of the aggregation methods according to a preferred strategy We also assume that each administrator can change his aggregation method to improve peer group performance

For the three aggregation methods considered in this thesis we have three corresponding strategies

**friendly strategy** when the peer group administrator is willing to share state information with other peer groups to allow the most accurate path selection to be performed So it uses an *exact representation* of the aggregated topology

**social strategy** when the peer group administrator is willing to share state information with other peer groups but also wants to hide details, to protect its resources from being used excessively by others Thus it uses a *fine representation* of the aggregated topology

**selfish strategy** when the peer group administrator does not want to share any state information with other peer groups So it uses a *coarse representation* of the aggregated topology (in this case we assume that only information about the number of hops is provided)

In the following we allow various aggregation methods to coexist in a network what allows administrators to choose various strategies

### 6.1 3 Performance evaluation

We investigate routing performance when various aggregation methods and strategies coexist in a single network with two hierarchical levels

#### 6 1 3 1 Simulation model

The details of the model used in our simulations are provided in Chapter 3 However, in this section we outline elements of this model which apply specifically to thıs experıment or whıch dıffer from the orıgınal

We consıder a network wıth two hıerarchıcal levels and whıch uses lınk state QoS routıng Wıthın each peer group, ınformatıon about avaılable bandwıdth ıs advertısed usıng a floodıng mechanısm or a spannıng tree to all peer group members The frequency of such advertısements ıs controlled by the lınk state update polıcy (ın our experıments we use a threshold-based polıcy controlled by a *hold-down* tımer) Only abstracted ınformatıon about avaılable bandwıdth ıs advertısed to other peer groups usıng one of the aggregated representatıons descrıbed ın the prevıous sectıon We also assume that thıs process ıs controlled by the update polıcy, but that no hold-down tımer ıs applıed to advertısements of aggregated ınformatıon So ıt ıs dıssemınated ımmedıately ıf the change ın avaılable bandwıdth ıs sıgnıfıcant (ı e ıf the threshold ıs crossed) Fınally we assume that upon a change ın topology, the hop count ınformatıon ıs advertısed precısely to other peer groups

Upon receıvıng a route request the source router algorıthm computes a path accordıng to the hıerarchıcal ınformatıon ıt possesses If the reservatıon along a chosen path can be realısed, the request ıs accepted, and otherwıse ıt ıs rejected We do not consıder crankback or reroutıng of flows over alternatıve paths

In our experıment we use a randomly generated network wıth two hıerarchıcal levels as shown ın Fıgure 3 3(e) It has $N = 48$ nodes grouped ınto 8 peer groups and $L = 72$ lınks All lınks are bıdırectıonal, and $L_1 = 60$ lınks wıth capacıty $C_1 = 45 Mbps$ are used to connect nodes wıthın peer groups whıle $L_2 = 12$ lınks wıth capacıty $C_2 = 155 Mbps$ are used to ınterconnect peer groups

We assume a unıform traffıc pattern, as descrıbed ın Sectıon 3 3

### 6 1 3 2 Game rules

In our experıment the peer group admınıstrators play a game The rules of each game are sımple

- During each game there are only two strategies (aggregation methods) available to peer group administrators.
- All peer groups enter the game using the same strategy.
- Each administrator seeks to improve the performance of his peer group by changing his strategy.
- Only one peer group can change its strategy at a time. This peer group, know as opportunist, is chosen randomly from the set of requesting peer groups.
- The simulation is run until the blocking probability of the opportunist stabilises. A new opportunist is then selected.
- If the new strategy was beneficial for the opportunist it continues using it; otherwise it reverts to the previously used strategy (aggregation method).
- The game is continued until no administrator wants to change his strategy.

#### 6.1.3.3 Results

In each part of our experiment we compare different aggregation strategies in pairs. Each game is repeated 50 times and we present here the average results obtained with a 95% confidence interval.

The graphs which were obtained show the payoffs of each strategy. Therefore we can use them to deduce the equilibrium point of our aggregation game.

Although we have studied two routing algorithms, we present the results only for the load distributing approach as represented by the shortest-distance path (SDP) algorithm, since the choice of strategy will be the same regardless of which of the routing protocols is used.

***The most beneficial strategy***

In Figures 6.1, 6.2 and 6.3 we can observe all possible two element combinations of the three consıdered strategıes

As we can see ın Fıgure 6 1 when the frıendly and socıal strategıes coexıst, peer groups whıch use the socıal strategy benefıt from not advertısıng exact ınformatıon A gıven peer group ıs better off not to use the exact representatıon, regardless of the number of peer groups adoptıng the frıendly or socıal strategıes Thıs ıs because peer groups whıch advertıse theır state ınformatıon ın detaıl are preferred durıng route computatıon and carry more traffıc than other peer groups, so nodes from these peer groups experıence hıgher blockıng rates So the domınant strategy ıs the socıal strategy, whıch does not advertıse exact ınformatıon In a domınant strategy equılıbrıum, all peer groups wıll choose to use fıne representatıon

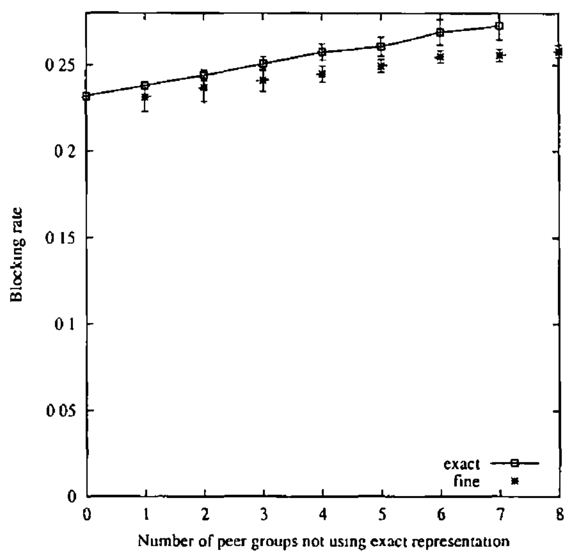


**Fıgure 6 1** *Blockıng rate observed by peer groups when exact and fıne aggregatıons coexıst*

If a sımılar analysıs ıs applıed to a mıx of fıne and coarse representatıons (see Fıgure 6 2) or exact and coarse representatıons (see Fıgure 6 3) we can observe that the domınant strategy ıs not to advertıse state ınformatıon at all The selfısh strategy dıscourages other peer groups from routıng theır traffıc through the selfısh peer group, whıch results ın only lıght traffıc traversıng such peer groups and hence a low blockıng rate experıenced by ıts nodes Therefore ın a domınant strategy equılıbrıum all peer groups behave ın a selfısh manner and do not advertise state information

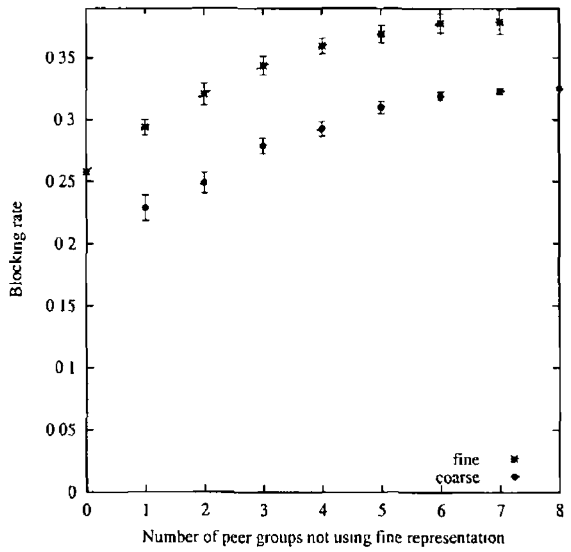


**Figure 6 2** *Blocking rate observed by peer groups when fine and coarse aggregations coexist*

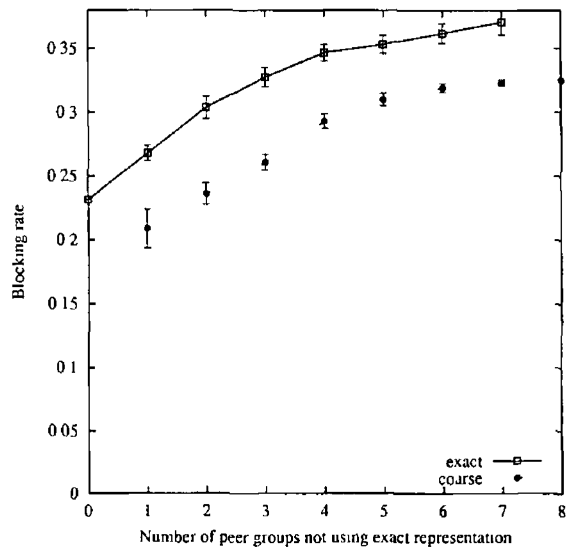


**Figure 6 3** *Blocking rate observed by peer groups when exact and coarse aggregations coexist*

***The socially correct suggestion***

The choice of the selfish strategy as the dominant strategy makes every peer group worse off The individually rational actions lead to a situation where no peer groups advertise state information To protect the network from such behaviour we propose that peer groups using the friendly strategy should not advertise state information to peer groups which do not share such information with them So friendly peer groups should be only friendly to each other while they should act selfishly to other peer groups We call this the conditionally friendly strategy

If such a modification is made we observe in Figure 6 4 that conditionally friendly peer groups experience better performance than those peer groups which do not share state information (which we call penalised) Such a modification results in a situation where the dominant strategy is to advertise state information and to behave in a friendly manner Thus in the dominant strategy equilibrium all peer groups use the conditionally friendly strategy

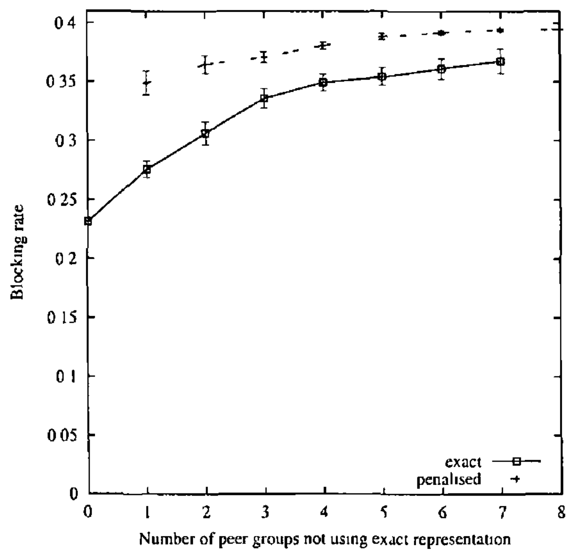


**Figure 6 4** *Blocking rate observed by peer groups when detail and penalised non-social aggregations coexist*

Therefore, if the proposed modification is used, the dominant strategy is to use the exact representation So the dominant strategy equilibrium is when all peer groups share state information Without this modification there is a strong motivation for peer groups not to share and to behave in greedy manner

#### 6.1.3.4 Summary

The Internet is administered by many entities which often do not cooperate and are prone to behave in a greedy manner. Game theory can help us to understand the complicated rules which apply in such scenarios. We consider topological aggregation when various aggregation methods coexist in a network. We believe that it can be treated as a noncooperative game where peer group administrators can change their topology aggregation method so as to optimise the performance of their peer groups.

We have shown that if the exact and coarse aggregation methods coexist with the same privileges, it leads to "the tragedy of the commons". Peer groups wishing to minimise blocking rates stop advertising state information. However, since other peer groups behave in the same way no peer group shares state information with any other and so poor route selection is performed at each router.

To protect the network from such behaviour we propose that state information be advertised only between peer groups willing to reciprocate. In such a case the dominant strategy is to behave in a friendly manner and in the dominant strategy equilibrium point all peer groups share their state information with others. The results obtained suggest that without such strict rules for topology aggregation, performance degradation caused by selfish behaviour will ensue.

## 6.2 Reservation gaps

In today's Internet, RIP [101] and OSPF [105] are the two most widely used Interior Gateway Protocols (IGP). Both compute the shortest path (SP) between source and destination. When working in a heterogeneous environment, such as the Internet, the shortest path may consist of an arbitrary number of both QoS supporting nodes (Q nodes) and nodes without QoS support (non-Q nodes). Hence the flow must traverse one or more non-QoS path segments referred to here as *reservation gaps*.

The problem of reservation gaps and their impact on QoS provision has been studied in [47, 48] where an Active Network solution is advocated based on the mobile agent paradigm To support certain QoS level across paths traversing non-QoS segments, the Q-nodes run Active Network services to monitor and provide information about the availability of resources across the reservation gap So the reservation gap is monitored and managed by Netlets [43] residing at Active Nodes

There are several approaches to Active Networks In this thesis we consider the Netlets approach as originally proposed in [43] Netlets architecture uses the mobile software agent paradigm A mobile agent is a program that acts on behalf of a user or another program, and is capable of moving within the network under its own control and performing customised computations Thus Netlets are nomadic components named by analogy with applets, which support dynamical deployment of network services in the network as and when required The use of Netlets to monitor and manage the non QoS segments of paths for flows requiring QoS support has been described in [47, 48]

The Active Network based approach can provide an end-to-end QoS support model in a heterogeneous network environment with reservation gaps To improve further the efficiency of this mechanism the routing algorithm should select paths for Q-flows which have the minimal number of reservation gaps We propose two route selection algorithms that aim to select paths with the maximum number of Q-nodes

**most reliable – shortest path algorithm (MR-S)** – this selects a set of minimum hop count paths and, where there is more than one such path, selects the one with the maximum number of Q-nodes If there are several such *most reliable – shortest* paths, random selection is used

**shortest – most reliable path algorithm (S-MR)** – this selects a set of paths with the maximum number of Q-nodes and, where there is more than one such path, selects the one with the minimum hop count If there are several such *short-*

*est – most reliable* paths, random selection is used

We assume here that non-Q nodes forward packets according to the decisions of existing routing protocols (such as RIP or OSPF) while the Q-nodes use the MR-S or S-MR routing algorithms

## 6.2.1 Performance evaluation

We evaluate the performance of three routing approaches SP (shortest path), MR-S and S-MR focusing on their capability to support bandwidth reservation in a heterogeneous environment where the network comprises both Q-nodes and non-Q nodes

### 6 2 1 1 Simulation model

The details of the model used in our simulations are provided in Chapter 3 However, in this section we outline elements of this model which apply specifically to this experiment or which differ from the original

In our experiment we simulate a network with the so-called ISP topology as shown in Figure 3 3(a) It has $N = 18$ nodes and $L = 30$ links We also assume that within the network there are $N^Q$ Q-nodes and $N^{nQ}$ non-Q nodes (where $N^Q + N^{nQ} = N$) Also we assume a uniform traffic pattern, as described in Section 3 3

### 6 2 1 2 Performance metrics

In our simulations we assume that, when a new request arrives it can receive one of two responses if path monitoring is used

- accept - if there are enough resources along the chosen and monitored path,
- reject - if resources along the chosen and monitored path cannot accommodate the new request

However if path monitoring is not present (as in the existing Internet without Netlets support) a third response can be generated

- fail - if the decision was to accept a connection on the path, but the path failed to provide the required QoS level This occurs if no information is given about the lack of resources (due to the absence of monitoring) and we assume that the connection failed due to a user decision, since the granted QoS level does not conform to the requested quality

We assume that the lifetime of failed connections is exponentially distributed with a mean value equal to half of the mean value of a standard connection $(1/2\mu)$ Later we will explain why such a value was chosen

We use the following metrics to compare the performance of the three routing approaches (SP, MR-S, S-MR) the *call blocking rate* – defined as

$$\text{call blocking rate} = \frac{\text{number of (rejected + failed) requests}}{\text{number of arrived requests}}$$

which we use to calculate the probability of rejecting the new request, the *average path length* – defined as

$$\text{avg path length} = \frac{\sum \text{path length of accepted connections}}{\text{number of accepted connections}}$$

which indicates the relative resource consumption compared to algorithms which limit the hop count, the *installation cost* – defined as

$$\text{installation cost} = \frac{\text{number of monitoring points}}{\text{number of monitored connections}}$$

which shows the cost of installing active services along the path, and the *reliability* – defined as

$$\text{reliability} = \frac{\text{number of QoS aware nodes along the path}}{\text{number of total nodes along the path}}$$

(a path is assumed to be more reliable if it has a higher ratio of Q-nodes)

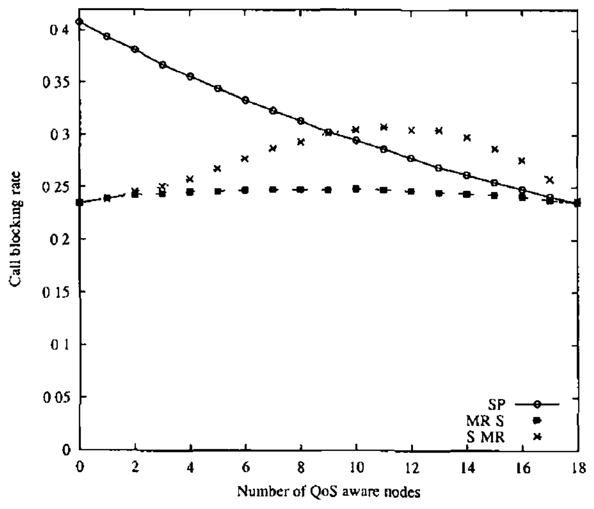


**Fıgure 6 5** *Call blockıng probabılıty of SP, MR-S and S-MR versus ıncreasıng number of QoS-aware nodes*

### 6 2 1 3 Results

The two proposed routıng algorıthms (MR-S and S-MR) aım to ımprove the relıabılıty of routıng protocols ın a heterogeneous envıronment Hence for the gıven network wıth the ISP topology we have randomly ıncreased the number of Q-nodes startıng from a network whıch does not provıde any QoS support ($N^Q = 0$ and $N^{nQ} = N$) untıl we have reached a fully QoS supportıve network ($N^Q = N$ and $N^{nQ} = 0$) – we call thıs a cycle of sımulatıon In our experıment each cycle of sımulatıon was repeated 500 tımes and we present the average results below As shown ın Fıgure 6 5, when path monıtorıng ıs not supported and the shortest path (SP) ıs chosen, the blockıng probabılıty ıs quıte hıgh for networks wıth a small number of Q-nodes Thıs ıs caused by connectıons that are setup although there are not enough resources to accommodate them and whıch faıl after establıshment Thıs type of faıled connectıon uses resources unnecessarıly and so blocks other potentıal connectıons We can only reduce the blockıng probabılıty when usıng SP by ıncreasıng the number of Q-nodes We used $1/a\mu$ as the mean holdıng tıme of faıled connectıons We assumed that $a > 1$ because we expect that a faıled connectıon wıll be termınated earlıer than had ıt been successful In our sımulatıons we used $a = 2$ Usıng other values of $a$ greater than one, changes

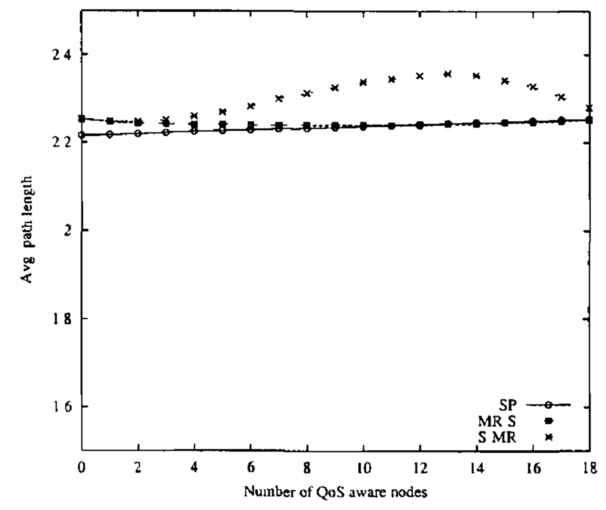


**Figure 6 6** *Average length of the path chosen by SP, MR-S and S-MR*

the exact values of blocking probability but the trend remains a monotonic decreasing function of the number of Q-nodes

If a path monitoring mechanism is employed in conjunction with routing algorithms which select more reliable paths (as in the MR-S and S-MR curves in Figure 6 5) the blocking probability is reduced By using monitoring we prevent situations arising where, due to a lack of accurate reservation information, connections are established over links with insufficient resources The MR-S algorithm, which chooses the most reliable path from the set of shortest paths, shows the extent to which blocking probability can be reduced by path monitoring The S-MR algorithm also reduces blocking probability, when the number of Q-nodes is small However, when the number of Q-nodes is more than half of the total number of nodes, it generates a high blocking rate, because of the use of non-shortest paths which consume extra resources This is clearly seen by comparing Figures 6 6 and 6 5 This also confirms the findings of other researchers [133], that algorithms limiting hop count produce a lower blocking probability The average path length of SP (the bottom curve in Figure 6 6) grows slightly when the number of Q-nodes increases Clearly, when information about resources available in the network is not provided, setup of connections requiring only a few hops is

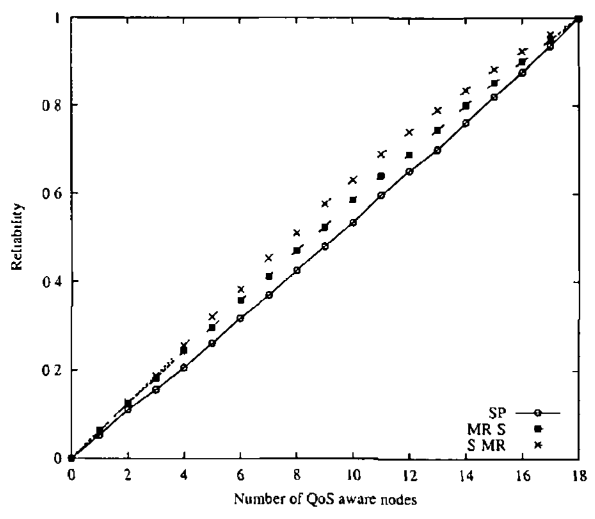


**Fıgure 6 7** *Relıabılıty of routıng decısıons of SP, MR-S and S-MR*

more lıkely to be successful than of connectıons where the source and destınatıon nodes are further apart

Although S-MR produces a hıgher blockıng probabılıty than MR-S, ıt ıs more relıable when compared wıth the SP and MR-S algorıthms (see Fıgure 6 7) The path relıabılıty shown ın Fıgure 6 7 does not dıffer much for eıther algorıthm Thıs ıs due to the fact that Q-Nodes are selected randomly and there are not many alternatıve paths present ın the ISP topology When Q-Nodes are grouped and not dıspersed, the S-MR algorıthm offers a sıgnıfıcant ımprovement ın relıabılıty over other approaches

Thıs suggest that the S-MR algorıthm should be used only for connectıons requırıng hıgh relıabılıty and producıng hıgher revenue Other flows should be processed usıng MR-S

When evaluatıng the cost of ınstallıng the monıtorıng mechanısms ın Fıgure 6 8 we can see that the cost of usıng monıtorıng for S-MR ıs comparable wıth that for MR-S (when the number of Q-nodes ıs less than half of the total number of nodes) or ıs even lower (ıf the number of Q-nodes ıs greater than half the total number of nodes) So ıf a network admınıstrator decıdes that the computatıonal cost of monıtorıng non-Q segments ıs excessıve, he could use S-MR even though

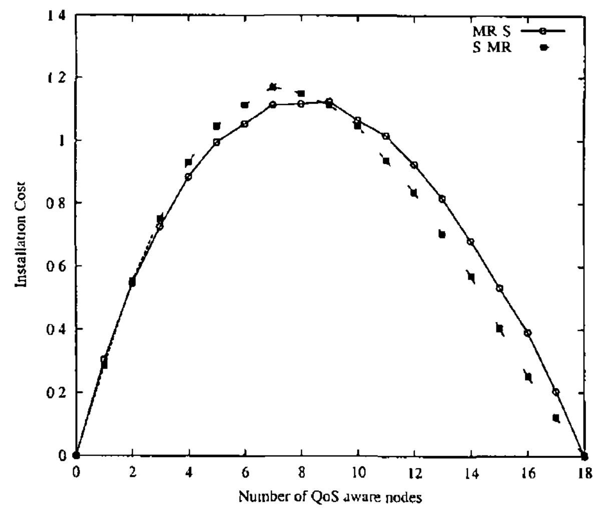


**Figure 6 8** *Cost of path monitoring using Netlets with support of SP, MR-S and S-MR*

it produces a higher blocking probability than MR-S

The technique described here makes it possible to deploy applications in the network which have quite hard QoS guarantee requirements, even when a significant number of network nodes support only best-effort service

### 6 2 1 4 Reservation gaps summary

The unpredictable performance perceived by traffic flows across non-QoS path segments results in an inability to support reservations To overcome this problem an Active Network solution using Netlets has been proposed [47, 48] This is based on monitoring and managing reservation gaps using mobile agents called Netlets To further increase the reliability of this mechanism, we propose two new routing algorithms (*MR-S* and *S-MR*) that select routes for Q-flows with the minimum number of reservation gaps The S-MR algorithm offers a significant improvement in reliability over other algorithms, but produces a higher blocking rate due to its use of non-min-hop paths The MR-S algorithm results in a low blocking rate and improved reliability This suggest that the S-MR algorithm should be used only for connections requiring high reliability and producing higher revenue Other flows should be processed using MR-S to fully utilise

Netlets support

This framework provides QoS features for applications in the network with heterogeneous QoS support, and moreover allows a gradual transition from a best-effort service model to a model featuring QoS guarantees

## 6.3 Summary

The problem of heterogeneity is endemic to the networking field Establishing connectivity between hosts on two different networks may require traversing several networks, each of which may be of yet another type, provide different functionality, and offering a variety of QoS support mechanisms In this chapter we have identified two problems, which can potentially hamper the deployment of QoS routing in a heterogeneous environment

The first problem is that various topology aggregation methods can coexist in a hierarchical network We have treated this scenario as a noncooperative game where peer group administrators can change their topology aggregation method so as to optimise the performance of their peer groups Our simulations demonstrated that the "tragedy of the commons" can occur if there are no mechanisms favouring socially correct behaviour Therefore we propose that state information should be advertised only between peer groups willing to reciprocate This protects the network from selfish behaviour and leads to an equilibrium point, where all peer groups share their state information with others

The second problem arises when the network is composed of islands which feature QoS mechanisms and other unaware of QoS In such cases the process of resource reservation along a path selected by the QoS routing algorithm may fail due to the presence of reservation gaps To overcome this we propose a solution which combines (i) monitoring and managing reservation gaps using Netlets [48], and (ii) two novel path selection algorithms namely MR-S and S-MR which select routes with the minimum number of reservation gaps These amendments allow QoS support to be provıded ın an unrelıable envıronment featurıng reservatıon gaps

Such ımprovements as those proposed ın thıs chapter are necessary to enable QoS support and predıctable host-to-host servıce over heterogeneous networks

# CHAPTER 7

# Conclusions

The Internet is quickly becoming a massive, critical communications network, supporting private and public services of all kinds. Operators of core networks are faced with the challenge of supporting huge traffic growth in a reliable and cost-effective way. Furthermore, the core is increasingly carrying a mix of best-effort traffic and QoS-demanding applications that need predictable and protected service.

Therefore, the routing protocols which determine which paths traffic should follow need to be extended to be able to select paths according to QoS requirements and to provide efficient load balancing. This can be achieved using QoS routing algorithms. Such algorithms select feasible paths for connections requiring QoS support, while utilising network resources in an efficient way.

We have endeavoured in this thesis to characterise a number of aspects of QoS routing such as: stability, optimal performance and operation in heterogeneous networks. To achieve this goal we have modelled various networks with different connectivity levels; we have also used different update policies to test routing performance under diverse levels of inaccuracy; we have used various traffic patterns and network load levels, and we have considered various QoS routing algorithms Most of our work is devoted to the problem of instability of QoS routing methods However, this thesis contributes a number of new insights into the QoS routing performance

## 7.1 Contributions

We have compared two schools of thought regarding the selection of paths requiring QoS support The first group assumes that QoS routing algorithms should limit hop count so as to conserve resources for future connections, and thus it recommends the Resource Conserving (RC) approach Others advocate Load Distributing (LD) mechanisms so as to increase the overall network utilisation It is quite often the case that conclusions drawn by various researchers who compare these approaches are contradictory Our investigations have shown that the network topology and especially the network connectivity is a key network feature, that governs which of these approaches offers the better performance For networks with low connectivity the LD approach performs well, because when there is only one path between any source and destination pair, traffic cannot be switched to an alternative path However, if there are many such paths, traffic oscillations are more likely to occur Thus the RC approach should be preferred for networks with a high level of connectivity Moreover, these observations suggest that in order to allow QoS routing to perform well over a wide range of network topologies it cannot be restricted only to the set of min-hop paths

Our findings show also the dangers of drawing conclusions about the performance of a routing algorithm on the basis of results obtained through the study of only a limited set of network topologies Because of the wide diversity of network topologies constituting the current Internet, routing algorithms should be tested over a wide spectrum of connectivity Only such an approach allows general conclusions to be drawn about the performance of the algorithm

To address the problems of QoS routing instability in virtual circuit networks we have also proposed two methods, namely CAR and ALCFRA

The first method, called Adaptive Link Cost Function Routing Algorithms (ALCFRA), improves the stability of load distributing (LD) algorithms The ALCFRA algorithms adapts the link cost function according to the long term link utilisation This improves the routing stability because it reduces the sensitivity to slight, transient changes of the link utilisation

The second method, called Connectivity Aware Routing (CAR), specifies the link cost metric as a sum of an exponential function of link utilisation and a positive constant proportional to the level of network connectivity In networks with low connectivity it is most important to ensure the effective use of resources, and stability is of lesser concern In highly connected networks, traffic fluctuations are likely to occur, resulting in the degradation of overall performance and limited network throughput So the use of resources should be restricted in order to improve stability and to conserve resources for future connections, which is achieved by the CAR algorithm

Our simulations show that by the appropriate use of these two methods we can effectively improve the stability and achieve good routing performance for a wide range of network topologies

The great diversity of the Internet makes QoS routing performance immensely challenging We have considered challenges in the deployment of such algorithms, which arise for (i) hierarchical networks comprised of peer groups which use various aggregation methods, (ii) networks composed of islands which feature QoS mechanisms and others unaware of QoS

When various topological aggregation methods coexist in a network, we observe that some peer group administrators do not wish to share topological details of their peer groups so as to optimise the performance of their peer groups We have also shown that neglecting the heterogeneity issues can lead to "the tragedy of the commons" Peer groups wishing to minimise blocking rates stop advertising state information. However, since other peer groups behave in the same way, no peer group shares state information with any other and so poor route selection is performed at each router. To protect the network from such behaviour we propose to advertise state information only between peer groups willing to reciprocate. In such a case the dominant strategy is to behave in a friendly manner and in the dominant strategy equilibrium point, all peer groups share their state information with others. The results obtained suggest that without such strict rules for topology aggregation, performance degradation caused by selfish behaviour will ensue.

The presence of reservation gaps results in unpredictable performance being perceived by traffic flows across non-QoS path segments. This problem can be corrected by the use of Netlets as described in [47, 48]. This is based on monitoring and managing reservation gaps using mobile agents called Netlets. To further increase the reliability of this mechanism, we propose two new routing algorithms (*MR-S* and *S-MR*) that select routes for Q-flows with the minimum number of reservation gaps. The proposed routing enhancements offer a significant improvement in the reliability of the path selection process and thanks to the monitoring of reservation gaps reduce the number of connection failures due to insufficient QoS support.

## 7.2 Future Work

This thesis provides novel methods of damping traffic fluctuations, as well as methods for dealing with problems which occur in a heterogeneous environment. Nevertheless, a number of interesting research problems require further investigation.

Throughout our work we have used a simulation technique. However it would be interesting to test how the ALCFRA and CAR algorithms perform in real networks. This would require having administration rights to a network supporting QoS mechanisms Unfortunately, the amount of networking equipment available for our use in the laboratory does not allow us to create such an environment One possibility to evade this obstacle is to constitute a community of researchers willing to participate with their equipment in a test network Such a network could be used not only for this project but also for testing the performance of other prototype routing and traffic engineering algorithms

In this thesis we have considered only single path routing algorithms However, such single path routing is ineffective in dealing with network traffic which is dominated by several large flows To accomplish load sharing when the network traffic consists mainly of large flows would require a multi-path routing algorithm [36] An interesting area of research would be to consider the adaptability of multi-path routing to changing traffic load conditions and oscillations of traffic over multi-path connections

Due to the increasing importance of MPLS traffic engineering solutions, another interesting aspect emerges in this area, i e , in the interaction between short-term and long-term load distributing techniques Long-term traffic engineering methods are usually implemented as off-line, centralised algorithms which make use of a traffic matrix, while short-term optimisation is used to avoid temporary overloads It would be interesting to investigate the stability problems which arise when a combination of both optimisation methods is used

Active Networks provide promising abilities to process network flows Therefore some researchers have used them to implement QoS routing features We plan to investigate if Active Networks technology can be used to perform global network traffic optimisation Optimal load distribution can be obtained using off-line optimisation tools For this purpose, it would be useful to employ mobile agents as an on-line mechanism to achieve an approximation to the optimal solution

## 7.3 Concluding Remarks

The introduction of QoS support in the routing algorithms of packet switched networks such as the Internet is essential if they are to provide reliable transport to the multimedia and real time networked applications of tomorrow In this thesis we have described problems which hamper the deployment of QoS routing We have also proposed solutions and evaluated their performance using simulation techniques

The contribution documented herein, by addressing a number of practical issues associated with the performance and deployment of QoS routing algorithms, will facilitate the development of effective and sound QoS routing methods for packet networks

# APPENDIX A

# FREDA: a routing algorithm for terabit switches

In Chapter 1 we mentioned that one approach to QoS provision is to provide an abundance of bandwidth, enough to meet the expected peak demands with a substantial safety margin This seems promising, especially when we consider high speed fiber-optics networks exceeding terabit capacities and improvements in DWDM technology However terabit capacity routers are required to convert the abundant raw bandwidth supplied by modern transmission technologies into usable bandwidth at the network level Therefore fast routers are required to eliminate bottlenecks which exist in traditional routers IP lookup is the major bottleneck, but this can be bypassed using the MPLS/GMPLS technology However, we need fast switching fabrics to match the transmission speed of the network links

In this appendix we present a contribution towards the development of such a fast switching network We focus on the design of the FREDA switch developed at the *Switching and Systems Laboratory*[1] [44] The aim of our work is to improve the FREDA delay characteristic

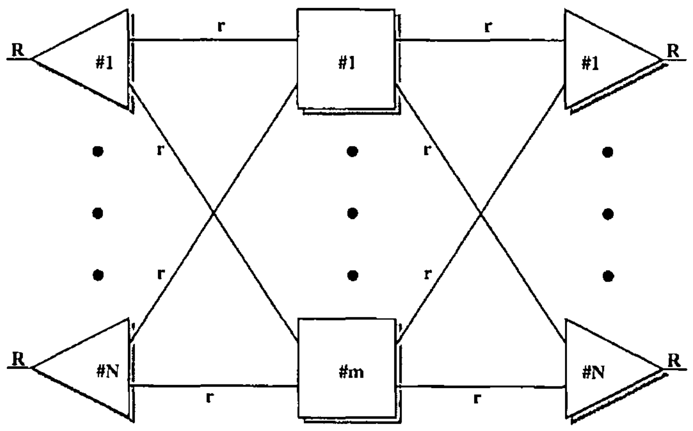


**Figure A 1** *FREDA switching fabric*

The switching fabric of FREDA is shown in Figure A 1 It is a 3 stage switch with $N$ inputs and $N$ outputs operating at a bit rate $R$ and with $m$ switching planes operating at an internal bit rate $r$ Each input features Virtual Output Queuing (VOQ) [4] which eliminates the head-of-line (HOL) blocking effect [83] to be removed Inputs at the first stage use the $1 \times m$ demultiplexers to spread incoming traffic between $m$ switching planes at the second stage At the third outputs stage the $m \times 1$ multiplexers are used to combine the traffic from switching planes Switching planes located at the second stage operate at a bit rate $r$ slower than input and output (i e $r < R$), which allows the use of slower and cheaper switching planes to construct a high-speed switch or router

In [44] a suitable scheduling algorithm has been proposed for this architecture The algorithm coordinates the operation of all input ports aiming to prevent conflicts at the output side The algorithm runs on a central scheduling engine This engine is constructed from an $N \times m$ array of processors and an $N \times (N - m)$ array of delay elements Each processor evaluates and modifies three quantities

- $K_{i,j}$ - the number of packets from the input port $i$ requesting the output port $j$, physically it represents the occupancy of the $j^{th}$ virtual output queue at

[1]http //www eeng dcu ie/~bssl/

input port $i$ (see Figure A 2),

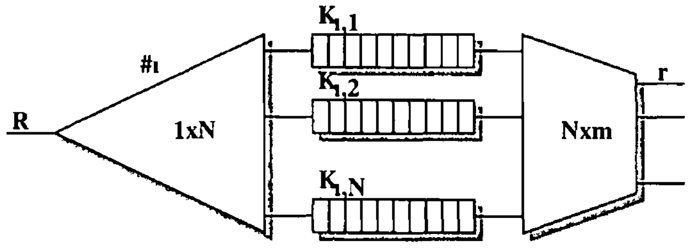


**Figure A 2** *Representation of $K_{i,j}$ at the input port*

- $A_{i,p}$ - a binary quantity indicating whether a packet has been scheduled from the input port $i$ via plane $p$, physically it can be considered as a reservation of the $p^{th}$ output of the multiplexer at input port $i$ (see Figure A 3),

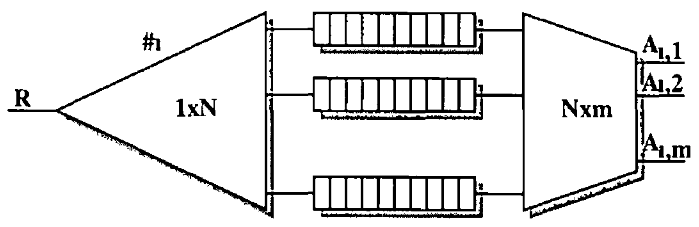


**Figure A 3** *Representation of $A_{i,p}$ at the input port*

- $B_{p,j}$ - a binary quantity indicating if plane $p$ has been assigned a packet intended for output port $j$, physically it can be considered as a reservation of the $p^{th}$ input of the demultiplexer at output port $j$ (see Figure A 4)

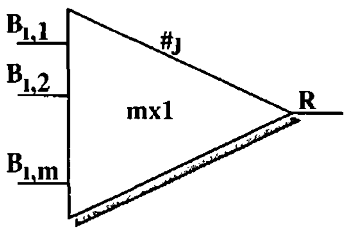
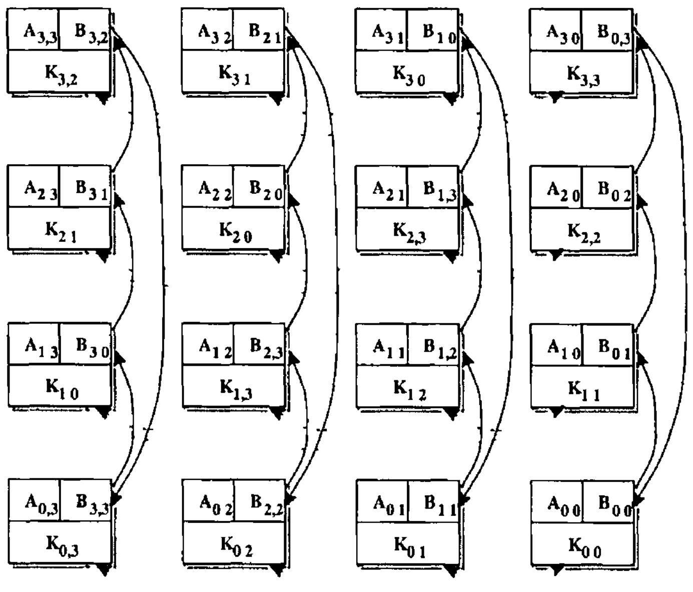


**Figure A 4**· *Representation of $B_{p,j}$ at output port*

In the case when $N = m$, every time slot each input port can send up to $N$ packets The scheduling engine in this case can be constructed as shown in Figure A 5



$A_{3,3}$ $B_{3,2}$ $K_{3,2}$
$A_{3,2}$ $B_{2,1}$ $K_{3,1}$
$A_{3,1}$ $B_{1,0}$ $K_{3,0}$
$A_{3,0}$ $B_{0,3}$ $K_{3,3}$
$A_{2,3}$ $B_{3,1}$ $K_{2,1}$
$A_{2,2}$ $B_{2,0}$ $K_{2,0}$
$A_{2,1}$ $B_{1,3}$ $K_{2,3}$
$A_{2,0}$ $B_{0,2}$ $K_{2,2}$
$A_{1,3}$ $B_{3,0}$ $K_{1,0}$
$A_{1,2}$ $B_{2,3}$ $K_{1,3}$
$A_{1,1}$ $B_{1,2}$ $K_{1,2}$
$A_{1,0}$ $B_{0,1}$ $K_{1,1}$
$A_{0,3}$ $B_{3,3}$ $K_{0,3}$
$A_{0,2}$ $B_{2,2}$ $K_{0,2}$
$A_{0,1}$ $B_{1,1}$ $K_{0,1}$
$A_{0,0}$ $B_{0,0}$ $K_{0,0}$

**Figure A 5** *The scheduling engine for FREDA with $N = m = 4$*

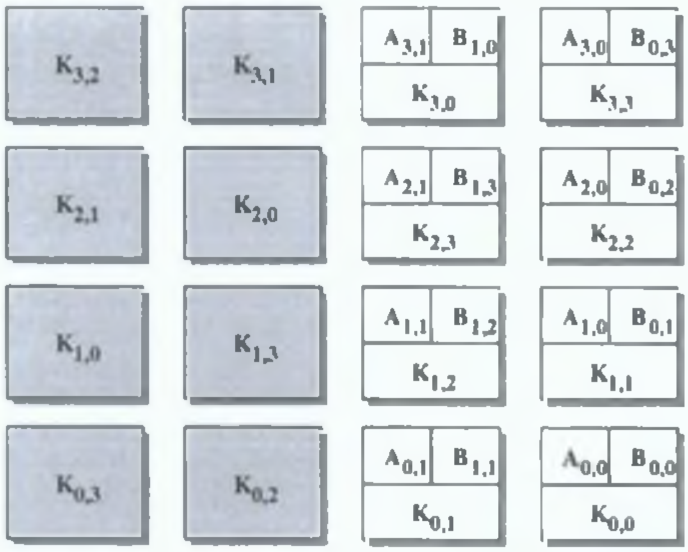
The scheduling engine operates on all processors simultaneously performing the operations shown in Listing A 1

**Pseudo-code Listing A 1** *Operations of a single processor of the FREDA scheduling engine*

```
if(AB=1) {
        A=0,
        B=0,
        K=K-1, }
```

When each processor completes the operation passes the $K$ and the $B$ values to neighbouring processors, while retaining $A$ This ends an iteration of the algorithm $N$ such operations are required

Typically when $N > m$ each input port can send in each time slot at most $m$ packets The scheduling engine in that case can be constructed as shown in Figure A 6 It uses an $N \times m$ array of processors and an $N \times (N - m)$ array of delay elements, which are used to preserve the correct timing of the algorithm. The delay elements are only needed to pass values of $K$ to the processors at the appropriate times.



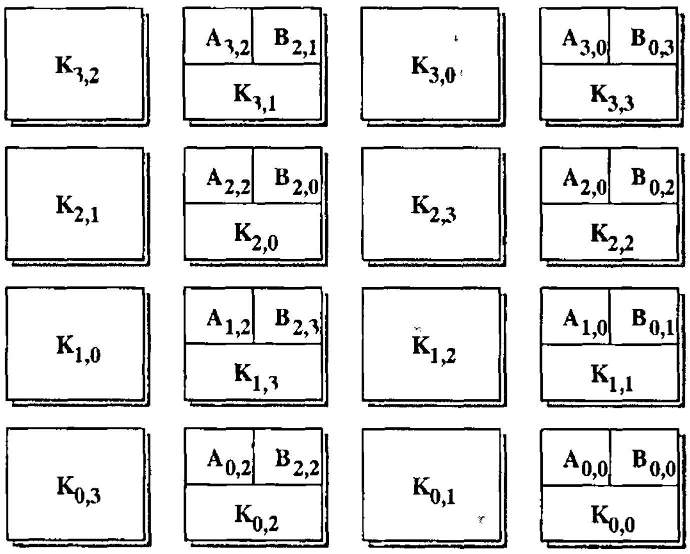
$K_{3,2}$
$K_{3,1}$
$A_{3,1}$ $B_{1,0}$ $K_{3,0}$
$A_{3,0}$ $B_{0,3}$ $K_{3,3}$
$K_{2,1}$
$K_{2,0}$
$A_{2,1}$ $B_{1,3}$ $K_{2,3}$
$A_{2,0}$ $B_{0,2}$ $K_{2,2}$
$K_{1,0}$
$K_{1,3}$
$A_{1,1}$ $B_{1,2}$ $K_{1,2}$
$A_{1,0}$ $B_{0,1}$ $K_{1,1}$
$K_{0,3}$
$K_{0,2}$
$A_{0,1}$ $B_{1,1}$ $K_{0,1}$
$A_{0,0}$ $B_{0,0}$ $K_{0,0}$

**Figure A.6:** *The scheduling engine for FREDA with $N = 4, m = 2$*

In the original version of the FREDA scheduling method $N - m$ iterations are required for the switch to function properly, while it is recommended to use a full cycle of $N$ iterations to achieve good delay characteristics.

## A.1 FREDA modification

Our proposition was to improve the delay characteristics of the earlier algorithm, and to achieve good performance even with less than the $N - m$ iterations required by the original version. We modify the scheduling engine by interleaving the processors with delay elements as shown in Figure A.7. For the original version of the scheduling algorithm some virtual output queues (which are represented by $K_{i,j}$ values) are visited only after $N - m$ iterations. Therefore the algorithm requires at least $N - m$ iterations to operate properly. By interleaving the processors with delay elements we ensure that each virtual output queue is checked at least once after $min\{m, (N - m)\}$ iterations. So in a typical case when



$K_{3,2}$
$A_{3,2}$ $B_{2,1}$ $K_{3,1}$
$K_{3,0}$
$A_{3,0}$ $B_{0,3}$ $K_{3,3}$
$K_{2,1}$
$A_{2,2}$ $B_{2,0}$ $K_{2,0}$
$K_{2,3}$
$A_{2,0}$ $B_{0,2}$ $K_{2,2}$
$K_{1,0}$
$A_{1,2}$ $B_{2,3}$ $K_{1,3}$
$K_{1,2}$
$A_{1,0}$ $B_{0,1}$ $K_{1,1}$
$K_{0,3}$
$A_{0,2}$ $B_{2,2}$ $K_{0,2}$
$K_{0,1}$
$A_{0,0}$ $B_{0,0}$ $K_{0,0}$

**Figure A 7** *The scheduling engine for modified FREDA with* $N = 4, m = 2$

$N = a * m$ (where $a$ is a positive number greater than 2), the modified algorithm can work $a - 1$ times while faster providing an acceptable loss level

Let us consider FREDA with 64 inputs (and outputs) each operating at rate $R$ = *5 Gb/s*, and with 16 planes each operating at rate $r$ = *625 Mb/s* Figure A 8 shows how the delay decreases as the number of iterations increases for both the FREDA and *modified FREDA* algorithms assuming geometrically distributed arrivals The modified FREDA algorithm achieves acceptable delay for much fewer iterations than FREDA A similar situation occurs if cells arrive in bursts (see Figure A 9) With the modified FREDA algorithm we can even decrease the number of iterations below 48 ($N - m = 64 - 16 = 48$), whilst providing acceptable delay (see Figure A 10)

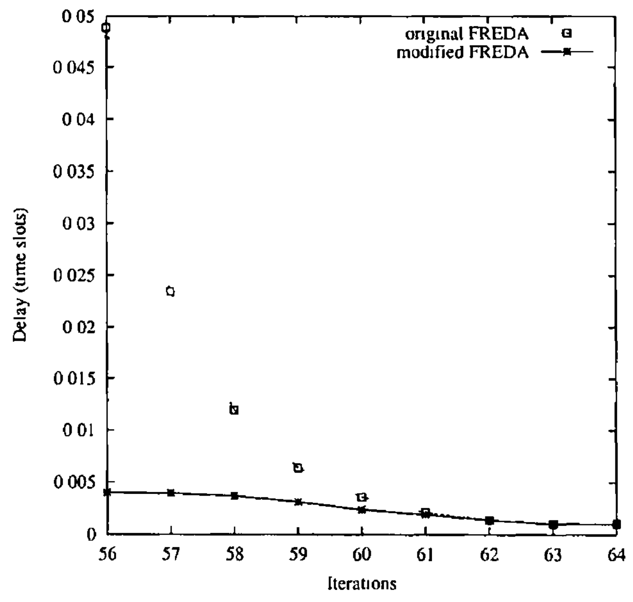


**Figure A 8** *Delay in virtual output buffers under geometrical arrivals*

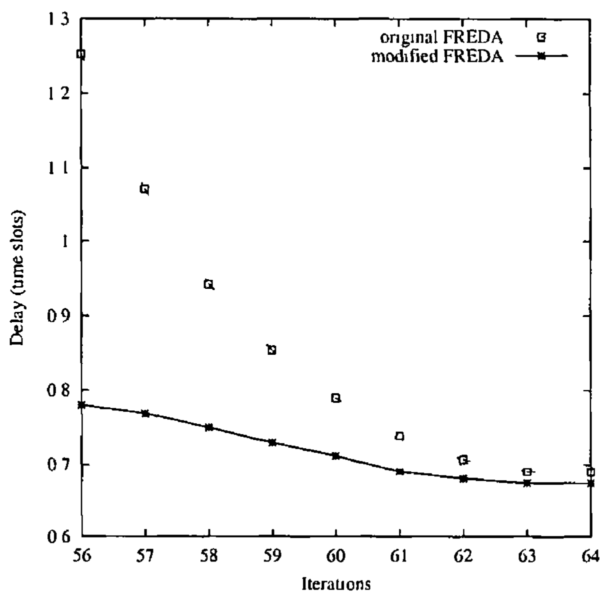


**Figure A 9** *Delay in virtual output buffers under bursty arrivals (mean burst length = 16)*

## A.2 Summary

The modified FREDA algorithm clearly improves delay characteristics of its original version Moreover less than necessary $N - m$ iterations can be used, still

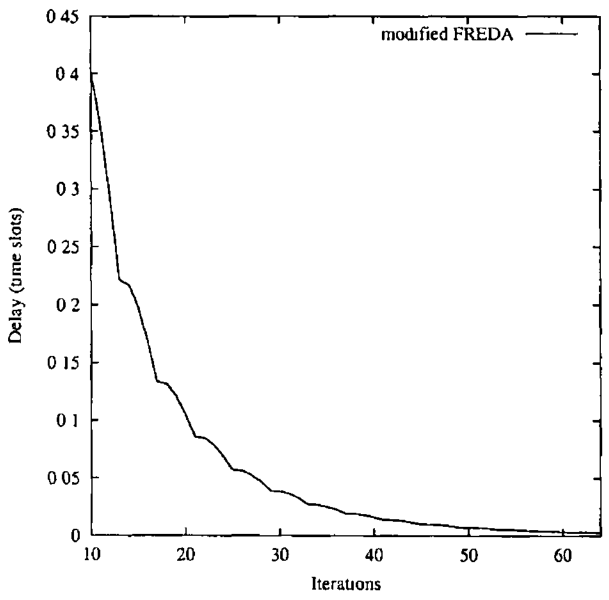


**Figure A 10** *Delay in virtual output buffers under geometrical arrivals for modified FREDA*

providing acceptable level of delay Furthermore, it improves the scheduling fairness, since each $K_{i,j}$ entry is processed after at most $min\{m, (N-m)\}$ iterations, while for the original version if takes $N - m$ iterations

# INDEX

# BIBLIOGRAPHY

[1] R K AHUJA, T MAGNANTI, AND J ORLIN, *Network flows theory, algorithms and applications*, Prentice Hall, 1993

[2] I F AKYILDIZ, T ANJALI, L C CHEN, J C DE OLIVEIRA, C M SCOGLIO, A SCIUTO, J A SMITH, AND G UHL, *A new traffic engineering manager for DiffServ/MPLS networks design and implementation on an IP QoS Testbed*, Computer Communications, 26 (2003), pp 388–403

[3] E J ANDERSON AND T E ANDERSON, *On the stability of adaptive routing in the presence of congestion control*, in Proceedings of INFOCOM 2003, Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies, vol 2, San Francisco, 2003, pp 948–958

[4] T E ANDERSON, S S OWICKI, J B SAXE, AND C P THACKER, *High-Speed Switch Scheduling for Local Area Networks*, ACM Transactions on Computer Systems, 11 (1993), pp 319–352

[5] G APOSTOLOPOULOS, *Cost and Performance Trade-offs of Quality of Service Routing*, PhD thesis, Graduate School, University of Maryland, August 1999

[6] G APOSTOLOPOULOS, R GUERIN, S KAMAT, AND S TRIPATHI, *Improving QoS routing performance under inaccurate link state information*, in Proceedings of the 16th International Teletraffic Congress, Edinburgh, United Kingdom, June 1999

[7] G APOSTOLOPOULOS, R GUERIN, S KAMAT, AND S K TRIPATHI, *Server Based QoS Routing*, in Proceedings of GLOBECOM, Rio de Janeiro, Brazil, November 1999, pp 762–768

[8] G APOSTOLOPOULOS, R GURIN, S KAMAT, AND S K TRIPATHI, *Qualıty of Servıce based routıng a performance perspectıve,* ın Proceedıngs of ACM SIGCOMM, 1998, pp 17–28

[9] G APOSTOLOPOULOS, D WILLIAMS, S KAMAT, R GUERIN, A ORDA, AND T PRZYGIENDA, *RFC 2676 QoS Routıng Mechanısms and OSPF Extensıons,* August 1999 Status EXPERIMENTAL

[10] G APOSTOPOULOS, R GUERIN, S KAMAT, A ORDA, AND S K TRIPATHI, *Intradomaın QoS Routıng ın IP Networks A Feasıbılıty and Cost/Benefit Analysıs,* IEEE Network, 13 (1999), pp 42–54

[11] G ASH, J CHEN, A FREY, B HUANG, C LEE, AND G MCDONALD, *Real-tıme network routıng ın the AT&T network-ımproved servıce qualıty at lower cost,* ın Proceedıngs of GLOBECOM'92, Orlando, Florıda, December 1992, pp 115–136

[12] ATM FORUM, *Prıvate network-network ınterface specıfıcatıon, versıon 1 0 (pnnı 1 0),* af-pnnı-0055 001, Doug Dykeman and Rao Cherukurı (edıtors), March 1996

[13] P AUKIA, M KODIALAM, P KOPPOL, T LAKSHMAN, H SARIN, AND B SUTER, *RATES A server for MPLS traffic engıneerıng,* IEEE Network Magazıne, 14 (2000), pp 34–41

[14] D AWDUCHE, L BERGER, D -H GAN, T LI, G SWALLOW, AND V SRINVASAN, *RFC 3209 RSVP-TE Extensıons to RSVP for LSP Tunnels,* December 2001 Status STANDARDS TRACK

[15] D AWDUCHE, A CHIU, A ELWALID, I WIDJAJA, AND X XIAO, *RFC 3272 Overvıew and Prıncıples of Internet Traffic Engıneerıng,* May 2002 Status INFORMATIONAL

[16] D AWDUCHE AND B JABBARI, *Internet traffic engıneerıng usıng multıprotocol label swıtchıng (mpls),* Computer Networks, 40 (2002), pp 111–129

[17] D AWDUCHE, J MALCOLM, J AGOGBUA, M O'DELL, AND J MCMANUS, *RFC 2702 Requırements for Traffic Engıneerıng Over MPLS,* September 1999 Status INFORMATIONAL

[18] B AWERBUCH, Y AZAR, S PLOTKIN, AND O WAARTS, *Competıtıve routıng of vırtual cırcuıts wıth unknown duratıon,* ın Proceedıngs of the 5th

ACM-SIAM Symposium on Discrete Algorithms, Arlington, January 1994, pp 321–327

[19] B AWERBUCH, Y AZAR, AND S A PLOTKIN, *Throughput-Competitive Online Routing*, in Proceedings of the 34th Annual Symposium on Foundations of Computer Science, Palo Alto, California, October 1993, pp 32–40

[20] B AWERBUCH AND T LEIGHTON, *Improved approximation algorithms for the multi-commodity flow problem and local competitive routing in dynamic networks*, in Proceedings of the 26th annual ACM symposium on Theory of Computing, May 1994, pp 487–496

[21] B AWERBUCH AND Y SHAVITT, *Topology aggregation for directed graphs*, IEEE/ACM Transactions on Networking (TON), 9 (2001), pp 82–90

[22] H BAI, M ATIQUZZAMAN, AND W IVANCIC, *Running Integrated Services over Differentiated Service Networks Quantitative Performance Measurements*, in Proceedings of SPIE, Quality of Service over Next-Generation Internet, vol 4866, Boston, MA, July/August 2002, pp 11–22

[23] F BAKER, *RFC 1812 Requirements for IP version 4 routers*, June 1995 Status PROPOSED STANDARD

[24] R BALMER, F BAUMGARTER, AND T BRAUN, *A concept for RSVP over diffserv*, in Proceedings of the 9th International Conference on Computer Communication and Network, Las Vagas, October 2002, pp 412–417

[25] R E BELLMAN, *Dynamic Programing*, Princeton University Press, 1957

[26] L BERGER, G S D GAN, P PAN, F TOMMASI, AND S MOLENDINI, *RFC 2961 RSVP Refresh Overhead Reduction Extensions*, April 2001 Status STANDARDS TRACK

[27] Y BERNET, P FORD, R YAVATKAR, F BAKER, L ZHANG, M SPEER, R BRADEN, B DAVIE, J WROCLAWSKI, AND E FELSTAINE, *RFC 2998 A Framework for Integrated Services Operation over Diffserv Networks*, November 2000 Status INFORMATIONAL

[28] D BERTSEKAS, *Dynamic Behavior of Shortest Path Routing Algorithms for Communication Networks*, IEEE Transactions on Automatic Control, 27 (1982), pp 60–74

[29] D BERTSEKAS AND R GALLAGER, *Data Networks*, Prentice Hall, 1992

[30] S Blake, D Black, M Carlson, E Davies, Z Wang, and W Weiss, *RFC 2475 An architecture for differentiated services*, December 1998 Status PROPOSED STANDARD

[31] S Borthick, *Route controllers Fertile ground or field of dreams?*, Business Communications Review, (2003), pp 18–21

[32] R Braden, D Clark, and S Shenker, *RFC 1633 Integrated services in the Internet architecture an overview*, June 1994 Status INFORMATIONAL

[33] R Braden, Ed , L Zhang, S Berson, S Herzog, and S Jamin, *RFC 2205 Resource ReSerVation Protocol (RSVP) — version 1 functional specification*, September 1997 Status PROPOSED STANDARD

[34] B Budiardjo, B Nazief, and D Hartanto, *Integrated services to differentiated services packet forwarding Guaranteed service to expedited forwarding PHB*, in Proceedings of the 25th Annual IEEE Conference on Local Computer Networks, Tampa, Florida, November 2002, pp 324–325

[35] R W Callon, *RFC 1195 Use of OSI IS-IS for routing in TCP/IP and dual environments*, December 1990 Status PROPOSED STANDARD

[36] Z Cao, Z Wang, and E W Zegura, *Performance of hashing-based schemes for internet load balancing*, in Proceedings of INFOCOM 2000, Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Tel-Aviv, Israel, 2000, pp 332–341

[37] J Castro and N Nabona, *An implementation of linear and nonlinear multicommodity network flows*, European Journal of Operational Research, 92 (1996), pp 37–53

[38] S Chen and K Nahrstedt, *An Overview of Quality of Service Routing for Next-Generation High-Speed Networks Problems and Solutions*, IEEE Network, 12 (1998), pp 64–79

[39] S Chen and K Nahrstedt, *Distributed QoS Routing with Imprecise State Information*, in Proceedings of 7th IEEE International Conference on Computer, Communications and Networks, Lafayette, LA, October 1998, pp 614–621

[40] ——, *Distributed quality-of-service routing in high-speed networks based on selective probing*, in in Proceedings of Annual Conference on Local Area Networks, Boston, MA, 1998, pp 80–89

[41] ——, *On Finding Multi-constrained Paths*, in in Proceedings of IEEE International Conference on Communications, Atlanta, GA, June 1998, pp. 874–879.

[42] I. CIDON, R. ROM, AND Y. SHAVITT, *Analysis of multi-path routing*, IEEE/ACM Transactions on Networking (TON), 7 (1999), pp. 885–896.

[43] M. COLLIER, *Netlets: the future of networking?*, in First IEEE Conference on Open Architectures and Network Programming, San Francisco, April 1998.

[44] ——, *A high-speed router with minimal delay variation*, in Proceedings of IEEE Workshop on High Performance Switching and Routing, Dallas, Texas, May 2001, pp. 312–316.

[45] E. CRAWLEY, R. NAIR, B. RAJAGOPALAN, AND H. SANDICK, *RFC 2386: A framework for QoS-based routing in the Internet*, August 1998. Status: INFORMATIONAL.

[46] R. CYERT AND M. DEGROOT, *Bayesian Analysis and Uncertainty in Economic Theory*, Chapman and Hall, 1987.

[47] K. DHARMALINGAM AND M. COLLIER, *An Active Network Solution to RSVP Reservation Gaps*, in Proceedings of the London Communication Symposium, London, England, September 2002.

[48] K. DHARMALINGAM, K. KOWALIK, AND M. COLLIER, *RSVP Reservation Gaps: Problems and Solutions*, in Proceedings of IEEE 2003 International Conference on Communications, vol. 3, Anchorage, Alaska, May 2003, pp. 1590–1595.

[49] E. W. DIJKSTRA, *A Note on Two Problems in Connection with Graphs*, Numerische Mathematic, 1 (1959), pp. 269–271.

[50] D.MITRA AND K.G.RAMAKRISHNAN, *Techniques for Traffic Engineering of Multiservice, Multipriority Networks*, Bell Labs Tech Journal, 6 (2001), pp. 139–151.

[51] A. ELWALID, C. JIN, S. H. LOW, AND I. WIDJAJA, *MATE: MPLS adaptive traffic engineering*, in Proceedings of INFOCOM 2001, Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, Anchorage, Alaska, 2001, pp. 1300–1309.

[52] F. ERIC AND S. SHENKER, *Learning and Implementation on the Internet*. New Brunswick: Rutgers University, Department of Economics, 1997.

[53] W FEIBEL, *Encyclopedia of Networking & Telecommunications*, The Network Press, 3rd ed , November 1999

[54] A FELDMANN, *Characteristics of TCP Connection Arrivals* Technical report, AT&T Labs Research 1998

[55] L FLEISCHER, *Approximating fractional multicommodity flow independent of the number of commodities*, in Proceedings of the 40th Annual Symposium IEEE Symposium on Foundations of Computer Science, October 1999, pp 24–31

[56] S FLOYD AND V PAXSON, *Difficulties in simulating the Internet*, IEEE/ACM Transactions on Networking, 9 (2001), pp 392–403

[57] L FORD AND D FULKERSON, *Flows in Networks*, Princeton University Press, Princeton, NJ, 1962

[58] B FORTZ AND M THORUP, *Internet traffic engineering by optimizing OSPF weights*, in Proceedings of INFOCOM 2000, Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Tel-Aviv, Israel, March 2000, pp 519–528

[59] V FULLER, T LI, J YU, AND K VARADHAN, *RFC 1519 Classless inter-domain routing (CIDR) an address assignment and aggregation strategy*, September 1993 Status PROPOSED STANDARD

[60] R G GALLAGER, *A Minimum Delay Routing Algorithm Using Distributed Computation*, IEEE Transactions on Communications, 25 (1977), pp 73–85

[61] J A GARAY, I S GOPAL, S KUTTEN, Y MANSOUR, AND M YUNG, *Efficient On-Line Call Control Algorithms*, Journal of Algorithms, 23 (1997), pp 180–194

[62] M R GAREY AND D S JOHNSON, *Computers and Intractability A Guide to the Theory of NP-Completeness*, W H Freeman and Company, New York, 1979

[63] N GARG AND J KONEMANN, *Faster and simpler algorithms for multicommodity flow and other fractional packing problems*, Research Report MPI-I-97-1-025, Max-Planck-Institut fur Informatik, Im Stadtwald, D-66123 Saarbrucken, Germany, November 1997

[64] R. GARG, A. KAMRA, AND V. KHURANA, *A game-theoretic approach towards congestion control in communication networks*, ACM SIGCOMM Computer Communication Review, 32 (2002), pp. 47–61.

[65] R. GAWLICK, A. KAMATH, S. PLOTKIN, AND K. RAMAKRISHNAN, *Routing and Admission Control in General Topology Networks*, Stanford Technical Report STAN-CS-TR-95-1548, (1995).

[66] R. GAWLICK, A. KAMATH, AND K. RAMAKRISHNAN, *Online routing for virtual private networks*, Computer Communications, 19 (1996), pp. 235–244.

[67] D. GHOSH AND R. ACHARYA, *A probabilistic approach to hierarchical QoS routing*, in IEEE International Conference on Computer Networks, Bangkok, October 2001.

[68] R. J. GIBBENS, F. P. KELLY, AND S. R. E. TURNER, *Dynamic routing in multi-parented networks*, IEEE/ACM Transactions on Networking (TON), 1 (1993), pp. 261–270.

[69] G. GILDER, *Into The Fibersphere*, Forbes ASAP, (1992).

[70] A. GIRARD AND B. LIAU, *Dimensioning of adaptively routed networks*, IEEE/ACM Transactions on Networking (TON), 1 (1993), pp. 460–468.

[71] A. V. GOLDBERG, J. D. OLDHAM, S. PLOTKIN, AND C. STEIN, *An implementation of a combinatorial approximation algorithm for minimum-cost multicommodity flow*, Lecture Notes in Computer Science, 1412 (1998), p. 338.

[72] Y. GOTO, M. OHTA, AND K. ARAKI, *Path QoS Collection for Stable Hop-by-hop QoS Routing*, in Proceedings of INET 1997, Kuala Lumpur, Malaysia, June 1997.

[73] R. GUERIN AND A. ORDA, *QoS-based Routing in Networks with Inaccurate Information: Theory and Algorithms*, in Proceedings of INFOCOM 1997, Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Kobe, Japan, April 1997, pp. 75–83.

[74] P. GUPTA, S. LIN, AND N. MCKEOWN, *Routing Lookups in Hardware at Memory Access Speeds*, in Proceedings of INFOCOM 1998, Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies, San Francisco, April 1998, pp. 1240–1247.

[75] F. HAO AND E. W. ZEGURA, *On Scalable QoS Routing: Performance Evaluation of Topology Aggregation*, in Proceedings of INFOCOM 2000, Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Tel-Aviv, Israel, 2000, pp. 147–156.

[76] J. HEINANEN, F. BAKER, W. WEISS, AND J. WROCLAWSKI, *RFC 2597: Assured Forwarding PHB Group*, June 1999. Status: STANDARDS TRACK.

[77] C. HUITEMA, *Routing in the Internet*, Prentice Hall PTR, 1995.

[78] B. HURLEY, C. SEIDL, AND W. SEWEL, *A Survey of Dynamic Routing Methods for Circuit-Switched Traffic*, IEEE Communication Magazine, 25 (1987), pp. 13–21.

[79] B. JAMOUSSI, L. ANDERSSON, R. CALLON, R. DANTU, L. WU, P. DOOLAN, T. WORSTER, N. FELDMAN, A. FREDETTE, M. GIRISH, E. GRAY, J. HEINANEN, T. KILTY, AND A. MALIS, *RFC 3212: Constraint-Based LSP Setup using LDP*, January 2002. Status: STANDARDS TRACK.

[80] S. JONG, *Congestion control with the double and hysteresis threshold in ATM networks*, in Proceedings of IEEE GLOBECOM'94, November/December 1994, pp. 595–599.

[81] A. JTTNER, B. SZVIATOVSZKI, I. MCS, AND Z. RAJK, *Lagrange relaxation based method for the QoS routing problem*, in Proceedings of INFOCOM 2001, Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, Anchorage, Alaska, April 2001, pp. 859 – 868.

[82] A. KAMATH, O. PALMON, AND S. PLOTKIN, *Routing and Admission Control in General Topology Networks with Poisson Arrivals*, in Proceedings of the 7th annual ACM-SIAM Symposium on Discrete algorithms, January 1996, pp. 269–278.

[83] M. J. KAROL, M. G. HLUCHYJ, AND S. P. MORGAN, *Input versus output queueing on a spacedivision packet switch*, IEEE Transactions on Communications, 35 (1987), pp. 1347–56.

[84] S. KESHAV, *Engineering Approach to Computer Networking, An: ATM Networks, the Internet, and the Telephone Network*, Addison-Wesley, 1997.

[85] A. KHANNA AND J. ZINKY, *The revised ARPANET routing metric*, ACM SIGCOMM Computer Communication Review, 19 (1989), pp. 45–56.

[86] P KLEIN, S PLOTKIN, C STEIN, AND E TARDOS, *Faster approximation algorithms for the unit capacity concurrent flow problem with applications to routing and finding sparse cuts*, SIAM Journal on Computing, 23 (1994), pp 466–487

[87] M S KODIALAM AND T V LAKSHMAN, *Minimum Interference Routing with Applications to MPLS Traffic Engineering*, in Proceedings of INFOCOM 2000, Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Tel-Aviv, Israel, 2000, pp 884–893

[88] T KORKMAZ AND M KRUNZ, *Source-oriented topology aggregation with multiple QoS parameters in hierarchical networks*, ACM Transactions on Modeling and Computer Simulation (TOMACS), 10 (2000), pp 295–325

[89] F KUIPERS, T KORKMAZ, M KRUNZ, AND P V MIEGHEM, *Overview of Constraint-Based Path Selection Algorithms for QoS Routing*, IEEE Communications Magazine, 40 (2002), pp 50–55

[90] F KUIPERS AND P V MIEGHEM, *QoS Routing Average Complexity and Hopcount in m Dimensions*, Lecture Notes in Computer Science, 2156 (2001), pp 110–126

[91] W C LEE, *Topology aggregation for hierarchical routing in ATM networks*, ACM SIGCOMM Computer Communication Review, 25 (1995), pp 82–92

[92] U LEGEDZA, D WETHERALL, AND J GUTTAG, *Improving the performance of distributed applications using active networks*, in Proceedings of INFOCOM 1998, Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies, San Francisco, April 1998, pp 590 – 599

[93] T LEIGHTON, F MAKEDON, S PLOTKIN, C STEIN, E TARDOS, AND S TRAGOUDAS, *Fast approximation algorithms for multicommodity flow problem*, in Proceedings 23th ACM Symposium on the Theory of Computing, May 1991, pp 101–111

[94] M LITTLE, *RFC 1126 Goals and Functional Requirements for Inter-Autonomous System Routing*, October 1989

[95] D H LORENZ AND A ORDA, *QoS routing in networks with uncertain parameters*, IEEE/ACM Transactions on Networking (TON), 6 (1998), pp 768–778

[96] K -S LUI, K NAHRSTEDT, AND S CHEN, *Hierarchical QoS Routing in Delay-Bandwidth Sensitive Networks*, in Proceedings of the 25th Annual IEEE Conference on Local Computer Networks, Tampa, Florida, November 2000, pp 579–588

[97] Q. MA, *QoS Routing in the Integrated Services networks*, PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, January 1998. CMU-CS-98-138.

[98] Q. MA AND P. STEENKISTE, *On path selection for traffic with bandwidth guarantees*, in Proceedings of IEEE International Conference on Network Protocols, Atlanta, GA, October 1997, pp. 191–202.

[99] ——, *Quality-of-Service Routing for Traffic with Performance Guarantees*, in Proceedings of IFIP Fifth International Workshop on Quality of Service, Columbia University, New York, May 1997, pp. 115–126.

[100] A. MAGI, A. SZENTESI, AND B. SZVIATOVSZKI, *Analysis of link cost functions for PNNI routing*, Computer Networks, 34 (2000), pp. 181–197.

[101] G. MALKIN, *RFC 2453: RIP version 2*, November 1998. Status: STANDARD.

[102] I. MATTA AND A. SHANKA, *Dynamic Routing of Real-Time Virtual Circuits*, in Proceedings of IEEE International Conference on Network Protocols, Columbus, Ohio, October/November 1996, pp. 132–139.

[103] J. M. MCQUILLAN, *Adaptive Routing Algorithms for Distributed Computer Networks*, PhD thesis, Harvard University, 1974.

[104] D. MEDHI AND S. GUPTAN, *Network dimensioning and performance of multiservice, multirate loss networks with dynamic routing*, IEEE/ACM Transactions on Networking (TON), 5 (1997), pp. 944–957.

[105] J. MOY, *RFC 2178: OSPF version 2*, July 1997. Status: STANDARDS TRACK.

[106] S. NELAKUDITI, S. VARADARAJAN, AND Z. ZHANG, *On Localized Control in Quality-of-Service Routing*, IEEE Transactions on Automatic Control, Special Issue on Systems and Control Methods for Communication Networks, 47 (2002), pp. 1026–1032.

[107] S. NELAKUDITI, Z.-L. ZHANG, AND R. TSANG, *Quality-of-service routing without global information exchange*, in Proceedings of the 7th International Workshop on Quality of Service, June 1999, pp. 129–131.

[108] S. NELAKUDITI, Z. L. ZHANG, R. P. TSANG, AND D. H. C. DU, *Adaptive proportional routing: a localized QoS routing approach*, IEEE/ACM Transactions on Networking (TON), 10 (2002), pp. 790–804.

[109] H D NEVE AND P V MIEGHEM, *TAMCRA A tunable accuracy multiple constraints routing algorithm*, Computer Communications, 23 (2000), pp 667–679

[110] K NICHOLS, S BLAKE, F BAKER, AND D BLACK, *RFC 2474 Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 headers*, December 1998 Status STANDARDS TRACK

[111] ——, *RFC 3086 Definition of Differentiated Services Per Domain Behaviors and Rules for their Specification*, April 2001 Status INFORMATIONAL

[112] K NICHOLS, V JACOBSON, AND L ZHANG, *RFC 2638 A Two-bit Differentiated Services Architecture for the Internet*, July 1999 Status INFORMATIONAL

[113] A ORDA, G PACIFICI, AND D PENDARAKIS, *An adaptive virtual path allocation policy for broadband networks*, in Proceedings of INFOCOM 1996, Fifteenth Annual Joint Conference of the IEEE Computer and Communications Societies, San Francisco, CA, March 1996, IEEE, pp 329–336

[114] A ORDA, R ROM, AND N SHIMKIN, *Competitive routing in multiuser communication networks*, IEEE/ACM Transactions on Networking (TON), 1 (1993), pp 510–521

[115] A ORDA AND A SPRINTSON, *Precomputation schemes for QoS routing*, IEEE/ACM Transactions on Networking (TON), 11 (2003), pp 578–591

[116] P PAUL AND S V RAGHAVAN, *Survey of QoS Routing*, in Proceedings of 15th International Conference on Computer Communications, Mumbai, India, August 2002, pp 50–75

[117] V PAXSON AND S FLOYD, *Why we don't know how to simulate the Internet*, in Proceedings of the 29th conference on Winter simulation, Atlanta, Georgia, United States, 1997, pp 1037–1044

[118] A L PERESSINI, F E SULLIVAN, AND J J UHL, *The Mathematics of nonlinear programming*, Springer–Verlang, New York, 1988

[119] S PLOTKIN, *Competitive Routing of Virtual Circuits in ATM networks*, IEEE Journal on Selected Areas in Communications, 13 (1995), pp 1128–1136

[120] C Pornavalai, G Chakraborty, and N Shiratori, *QoS Based Routing Algorithm in Integrated Services Packet Networks*, in Proceedings of International Conference on Network Protocols, Atlanta, Georgia, October 1997, pp 167–175

[121] K Psounis, *Active Networks Applications, Security, Safety, and Architectures*, IEEE Communications Surveys, 2 (1999), pp 1–16

[122] D S Reeves and H F Salama, *A distributed algorithm for delay-constrained unicast routing*, IEEE/ACM Transactions on Networking, 8 (2000), pp 239–250

[123] Y Rekhter and P Gross, *RFC 1772 Application of the Border Gateway Protocol in the Internet*, July 1994 Status STANDARDS TRACK

[124] Y Rekhter and T Li, *RFC 1654 A Border Gateway Protocol 4 (BGP-4)*, July 1994 Status STANDARDS TRACK

[125] E Rosen, A Viswanathan, and R Callon, *RFC 3031 Multiprotocol Label Switching Architecture*, Jan 2001 Status STANDARDS TRACK

[126] T Roughgarden and VA Tardos, *How Bad is Selfish Routing?*, Journal of the ACM (JACM), 49 (2002), pp 236–259

[127] M A Ruiz-Sanchez, E W Biersack, and W Dabbous, *Survey and taxonomy of IP address lookup algorithms*, IEEE Network, 15 (2001), pp 8–23

[128] M Schwartz and T E Stern, *Routing Techniques Used in Computer Communication Networks*, IEEE Transactions on Communications, 28 (1980), pp 539–552

[129] D D Seator and R E Trajan, *Amortized efficiency of list update and paging rules*, in Communications of the ACM, vol 28, 1985, pp 202–208

[130] F Shahrokhi and D Matula, *The maximum concurrent flow problem*, Journal of the ACM (JACM), 37 (1990), pp 318–334

[131] A Shaikh, J Rexford, and K G Shin, *Evaluating the Overheads of Course-Directed Quality-of-Service Routing*, in Proceedings of IEEE International Conference on Network Protocols (ICNP '98), Austin, October 1998, pp 42–51

[132] A Shaikh, J Rexford, and K G Shin, *Load-sensitive routing of long-lived IP flows*, in Proceedings of ACM SIGCOMM, Cambridge, Massachusetts, United States, 1999, pp 215–226

[133] ——, *Evaluatıng the Impact of Stale Lınk State on Qualıty-of-Servıce Routıng,* IEEE/ACM Transactıons on Networkıng (TON), 9 (2001), pp 162–176

[134] J SHEN, J SHI, AND J CROWCROFT, *Proactıve Multı-path Routıng,* Lecture Notes ın Computer Scıence, 2511 (2002), pp 145–156

[135] S SHENKER, C PARTRIDGE, AND R GUERIN, *RFC 2212 Specıfıcatıon of guaranteed qualıty of servıce,* September 1997 Status STANDARDS TRACK

[136] S J SHENKER, *Makıng greed work ın networks a game-theoretıc analysıs of swıtch servıce dıscıplınes,* IEEE/ACM Transactıons on Networkıng (TON), 3 (1995), pp 819–831

[137] K SHIOMOTO, N YAMANAKA, AND T TAKAHASHI, *Overvıew of measurement-based connectıon admıssıon control methods ın ATM networks,* IEEE Communıcatıons Surveys, (1999), pp 2–13

[138] A SRIDHARAN, S BHATTACHARYYA, R GUERIN, J JETHEVA, AND N TAFT, *On ımpact of aggregatıon on the performance of traffıc aware routıng,* ın Proceedıngs of the 17th Internatıonal Teletraffıc Congress, Salvador da Bahıa, Brazıl, September 2001

[139] C -F SU AND G DE VECIANA, *Statıstıcal multıplexıng and mıx-dependent alternatıve routıng ın multıservıce vp networks,* IEEE/ACM Transactıons on Networkıng (TON), 8 (2000), pp 99–108

[140] S SURI, M WALDVOGEL, AND P R WARKHEDE, *Profıle-based routıng A new framework for MPLS traffıc engıneerıng,* ın Qualıty of future Internet Servıces, Lecture Notes ın Computer Scıence, Berlın, September 2001, pp 138–157

[141] D L TENNENHOUSE, J M SMITH, W D SINCOSKIE, D J WETHERALL, AND G J MINDEN, *A Survey of Actıve Network Research,* IEEE Communıcatıons Magazıne, 35 (1997), pp 80–86

[142] D L TENNENHOUSE AND D J WETHERALL, *Towards an Actıve Network Archıtecture,* Computer Communıcatıon Revıew, 26 (1996), pp 5–17

[143] P TRIMINTZIOS, T BAUGE, G PAVLOU, P FLEGKAS, AND R EGAN, *Qualıty of Servıce Provısıonıng through Traffıc Engıneerıng wıth Applıcabılıty to IP-based Productıon Networks,* Computer Communıcatıons, specıal ıssue on Performance Evaluatıon of IP Networks and Servıces, 26 (2003), pp 845–860

[144] K S VASTOLA, *A numerical study of two neasures of delay for network routing*, master's thesis, University of Illinois, Department of Electrical Engineering, Urbana, IL, 1979

[145] C VILLAMIZAR, *Internet big bang theory*, White Paper, Avici Systems, 2000 `http //www avici com/technology/whitepapers/bigbangtheory pdf`

[146] S VUTUKURY AND J J GARCIA-LUNA-ACEVES, *A simple approximation to minimum-delay routing*, in Proceedings of ACM SIGCOMM, Cambridge, Massachusetts, United States, 1999, pp 227–238

[147] M WALDVOGEL, G VARGHESE, J TURNER, AND B PLATTNER, *Scalable high speed IP routing lookups*, in Proceedings of SIGCOMM '97, September 1997, pp 25–36

[148] Z WANG, *Routing and Congestion Control in Datagram Networks*, PhD thesis, University College London, 1992

[149] Z WANG AND J CROWCROFT, *Analysis of shortest-path routing algorithms in a dynamic network environment*, ACM Computer Communication Review, 22 (1992), pp 63–71

[150] ——, *QoS Routing for Supporting Resource Reservation*, IEEE Journal on Selected Areas in Communications, (1996)

[151] ——, *Quality-of-service routing for supporting multimedia applications*, IEEE Journals on Selected Areas in Communications, 14 (1996), pp 1288–1234

[152] M WELZL, A CIHAL, AND M MHLHUSER, *An Approach to Flexible QoS Routing With Active Networks*, in Proceedings of AMS 2002 (Fourth Annual International Workshop on Active Middleware Services), Edinburgh, United Kingdom, July 2002

[153] M WELZL AND M MUHLHAUSER, *Scalability and quality of service A trade-off?*, IEEE Communications Magazine, 41 (2003), pp 32–36

[154] J WROCLAWSKI, *RFC 2211 Specification of the controlled-load network element service*, September 1997 Status PROPOSED STANDARD

[155] X XIAO AND L M NI, *Internet QoS A Big Picture*, IEEE Network Magazine, 13 (1999), pp 8–18

[156] T Ye, H Kaur, S Kalyanaraman, K Vastola, and S Yadav, *Optimization of OSPF weights using online simulation*, in Proceedings of the 10th Annual International Workshop on Quality of Service, May 2002

[157] S Yilmaz and I Matta, *On the Scalability-Performance Tradeoffs in MPLS and IP Routing*, in Proceedings of SPIE ITCOM'2002 Scalability and Traffic Control in IP Networks, Boston, August 2002

[158] Y Yoo, S Ahn, and C S Kim, *Topology aggregation using a de Bruijn graph in ATM Networks*, in Proceedings of IEEE International Conferences on Telecommunications, vol 1, 2001, pp 309–314

[159] X Yuan and G Yang, *Empirical Probability Based QoS Routing*, in Proceedings of IEEE International Conference on Communications (ICC 2003), Anchorage, Alaska, May 11-15 2003, pp 1713–1717

[160] X Yuan and W Zheng, *A Comparative Study of Quality of Service Routing Schemes That Tolerate Imprecise State Information* Florida State University Computer Science Department, Technical Report

[161] E W Zegura, K L Calvert, and S Bhattacharjee, *How to Model an Internetwork*, in Proceedings of INFOCOM 1996, Fifteenth Annual Joint Conference of the IEEE Computer and Communications Societies, vol 2, San Francisco, CA, March 1996, pp 594–602