CORE
Metadata, citation and similar papers at core.ac.uk
Provided by Bradford Scholars

UNIVERSITY of BRADFORD

Library

University of Bradford eThesis

This thesis is hosted in Bradford Scholars – The University of Bradford Open Access repository. Visit the repository for full metadata or to contact the repository team

© University of Bradford. This work is licenced for reuse under a Creative Commons Licence.

LOCALIZED QUALITY OF SERVICE ROUTING ALGORITHMS FOR COMMUNICATION NETWORKS

The Development and Performance Evaluation of Some New Localized Approaches to Providing Quality of Service Routing in Flat and Hierarchical Topologies for Computer Networks

Ahmed S. Alzahrani

A thesis submitted for the degree of
Doctor of Philosophy

Department of Computing

University of Bradford

2009

Abstract

Quality of Service (QoS) routing considered as one of the major components of the QoS framework in communication networks. The concept of QoS routing has emerged from the fact that routers direct traffic from source to destination, depending on data types, network constraints and requirements to achieve network performance efficiency. It has been introduced to administer, monitor and improve the performance of computer networks. Many QoS routing algorithms are used to maximize network performance by balancing traffic distributed over multiple paths. Its major components include bandwidth, delay, jitter, cost, and loss probability in order to measure the end users' requirements, optimize network resource usage and balance traffic load. The majority of existing QoS algorithms require the maintenance of the global network state information and use it to make routing decisions. The global QoS network state needs to be exchanged periodically among routers since the efficiency of a routing algorithm depends on the accuracy of link-state information. However, most of QoS routing algorithms suffer from scalability problems, because of the high communication overhead and the high computation effort associated with marinating and distributing the global state information to each node in the network.

The goal of this thesis is to contribute to enhancing the scalability of QoS routing algorithms. Motivated by this, the thesis is focused on localized QoS routing that is proposed to achieve QoS guarantees and overcome the problems of using global network state information such as high communication overhead caused by frequent state information updates, inaccuracy of link-state information for large QoS state update intervals and the route oscillating due to the view of state information. Using such an approach, the source node makes its own routing decisions based on the information that is local to each node in the path. Localized QoS routing does not need the global network state to be exchanged among network nodes because it infers the network state and avoids all the problems associated with it, like high communication and processing overheads and oscillating behaviour. In localized QoS routing each source node is required to first determine a set of candidate paths to each possible destination.

In this thesis we have developed localized QoS routing algorithms that select a path based on its quality to satisfy the connection requirements. In the first part of the thesis a localized routing algorithm has been developed that relies on the average residual bandwidth that each path can support to make routing decisions. In the second part of the thesis, we have developed a localized delay-based QoS routing (DBR) algorithm which relies on a delay constraint that each path satisfies to make routing decisions. We also modify credit-based routing (CBR) so that this uses delay instead of bandwidth. Finally, we have developed a localized QoS

routing algorithm for routing in two levels of a hierarchal network and this relies on residual bandwidth to make routing decisions in a hierarchical network like the internet.

We have compared the performance of the proposed localized routing algorithms with other localized and global QoS routing algorithms under different ranges of workloads, system parameters and network topologies. Simulation results have indicated that the proposed algorithms indeed outperform algorithms that use the basics of schemes that currently operate on the internet, even for a small update interval of link state. The proposed algorithms have also reduced the routing overhead significantly and utilize network resources efficiently.

Acknowledgments

I am forever grateful to my supervisor, Professor Michael E. Woodward, whose personal involvement and generosity in time and attention have made this thesis possible. I have truly been privileged to have the honour to be one of his students, to learn from and work with one of the best in the field, and in the process, to have the opportunity to share the knowledge I learned from him with others. Prof. Woodward was the reason I became interested in the field of QoS routing. His humility and sense of humour can turn any curious student into a passionate convert. Despite all the challenges I have faced during the course of my graduate study, he has been the most sympathetic and understanding.

Any success I have attained during my academic career I attribute to my parents who encouraged me in every endeavor and taught me to value education. They have been a great support throughout my life.

Undiminished love and deepest gratitude is devoted to my loving wife and my two wholesome children, Osama and Tala. This thesis would not have prospered without the family's patience, support, faith, love and laughter.

Finally, I would like to thank the ministry of higher education in Saudi Arabia for granting me the scholarship and for providing full financial support during the whole period of my graduate studies.

List of publications

[1] A. Alzahrani, and M. E. Woodward, "Localized quality-of-service routing using delay", Proceedings of the 4th IEEE/IFIP International Conference on Internet, ICI 2008, pp.1-6, 23-25 Sept. 2008.

[2] A. Alzahrani, and M. E. Woodward, "Residual bandwidth as localized QoS routing metric", Proceedings of the 16th International Software, Telecommunications and Computer Networks, SoftCOM 2008, pp.125-129, 25-27 Sept. 2008.

[3] A. Alzahrani, and M. E. Woodward, "Path Selection Method for Localized QoS Routing", Proceedings of the New Technologies, Mobility and Security, NTMS '08, pp.1-5, 5-7 Nov. 2008.

[4] A. Alzahrani, and M. E. Woodward, "End-to-End delay in localized QoS routing", Proceedings of the 11th IEEE International Conference on Communication Systems, ICCS 2008, pp.1700-1706, 19-21 Nov. 2008.

[5] A. Alzahrani, and M. E. Woodward, "Localized QoS Routing in Hierarchical Networks", Submitted to *Elsevier Computer Networks*.

Table of Contents

Abstract ..... **ii**

Acknowledgments ..... **v**

List of publications ..... **vi**

Table of Contents ..... **vii**

List of Figures ..... **xiii**

List of Principal Abbreviations ..... **xv**

List of Principal Symbols ..... **xviii**

Chapter 1 Introduction **1**

1.1 Background 1

1.2 Motivation 3

1.3 Thesis Aims and Objectives 5

1.4 Thesis contributions 6

1.5 Thesis outline 8

Chapter 2 ..... **10**

QoS Routing **10**

2.1 Introduction 10

2.2 Internet Quality of Service: Mechanism and Architecture 12

2.2.1 Integrated Service (Intserv) [11] 13

2.2.2 Differentiated Services (DiffServ) [13] 14
2.2.3 Multiprotocol Label Switching (MPLS) [14] 15
2.2.4 Traffic Engineering (TE) and Constraint Based Routing (CBR) [22] 16
2.3 QoS Routing 17
2.3.1 Notations and Metrics 17
2.3.2 Maintenance of State Information 19
2.3.3 Update Frequency and Information Inaccuracy 21
2.3.4 QoS Routing Problems 23
2.3.4.1 The Unicast Routing Problem 23
Single constraint routing problems 24
Multiple constraint routing problem 25
2.3.4.2 The Multicast Routing Problem 25
2.3.4.3 NP-Completeness 26
2.3.5 QoS Routing Strategies 27
2.3.5.1 Source Routing 27
2.3.5.2 Distributed Routing 28
2.3.5.3 Hierarchical Routing 29
2.3.6 Routing in the Internet 30
Chapter 3 31
QoS Routing Algorithms 31

3.1 Introduction 31
3.2 Time Complexity 32
3.3 Shortest Path Algorithms 33
3.4 Global Routing Algorithms 34
3.5 Localized Routing Algorithms 38
Chapter 4 48
Simulation Model and Performance Parameters 48
4.1 Introduction 48
4.2 Graph Model 48
4.2.1 Random Topology 49
4.2.2 Regular Topology 51
4.2.3 Hierarchical Topology 52
4.2.4 ISP Topology 53
4.3 Simulation Environment 54
4.3.1 Simulator Design 54
4.3.2 Simulator structure 55
4.3.3 Performance metrics 61
4.3.3.1 Blocking probability 61
4.3.3.2 Load balancing 62
4.3.4 Simulator validation 63
4.4 Summary 65

Chapter 5 66

Localized Bandwidth Based QoS Routing 66

5.1 Introduction 66

5.2 Bandwidth Based Routing 68

5.3 Performance evaluation 72

5.3.1 Simulation model 72

5.3.2 Traffic generation 73

5.3.3 Performance metrics 75

5.3.4 Simulation results 76

5.3.5 Blocking probabilities 76

5.3.6 Impact of various load conditions 76

5.3.7 Impact of bursty traffic 78

5.3.8 Impact of large bandwidth 80

5.3.9 Impact of network topologies 81

5.3.10 BBR sensitivity to W parameter 84

5.3.11 BBR overhead and time complexity 85

5.4 Summary 88

Chapter 6 89

Localized Delay Based QoS Routing 89

6.1 Introduction 89

6.2 The proposed algorithms 90

6.2.1 Credit Based Routing 91
6.2.2 Delay Based Routing 96
6.3 Performance evaluation 100
6.3.1 Simulation model 101
6.3.2 Traffic generation 102
6.3.3 Performance metrics 103
6.3.4 Simulation results 104
6.3.5 Delay constraints 104
6.3.6 Impact of network topologies and varying arrival rates 109
6.3.7 Impact of varying non-uniform traffic 112
6.3.8 Impact of bursty traffic 115
6.3.9 Impact of heterogeneous delay constraints 117
6.3.10 DBR and CBR stability and sensitivity to their parameters 120
6.3.11 Load Balancing 125
6.3.12 DBR and CBR time complexity 127
6.4 Summary 128
Chapter 7 Localized QoS Routing in Hierarchical Networks 129
7.1 Introduction 129
7.2 The proposed algorithm 132
7.2.1 HBBR routing operation 133
7.2.2 HBBR algorithm 135

7.3 Performance evaluation 138
7.3.1 Simulation model 139
7.3.2 Traffic generation 144
7.3.3 Performance metrics 146
7.3.4 Simulation results 147
7.3.5 Blocking probabilities 147
7.3.6 Impact of network topologies 151
7.3.7 Impact of varying non-uniform traffic 153
7.3.8 Impact of bursty traffic 156
7.3.9 Impact of heterogeneous traffic 159
7.3.10 HBBR Stability 160
7.3.11 Load Balancing 165
7.3.12 HBBR sensitivity to W parameter 168
7.3.13 HBBR time complexity 169
7.4 Conclusions 170
Chapter 8 Conclusions and Future Work 171
8.1 Conclusions 171
8.2 Future Works 173
References 175

List of Figures

Figure 3.1 Flow diagram for DAR algorithm 39

Figure 3.2 Pseudo-code for PSR. 41

Figure 4.1 Random 40 topology 51

Figure 4.2 a 4-node torus and lattice graph 52

Figure 4.3 the ISP topology 53

Figure 4.4 Functional components localized simulator 55

Figure 4.5 Functional components Global simulator 57

Figure 4.6 Simulator Validation Results 64

Figure 5.1 Flow and bandwidth blocking probabilities in Random 40. 78

Figure 5.2 Impact of bursty traffic in Random 40. 79

Figure 5.3 Impact of large bandwidth in Random 80. 81

Figure 5.4 Impact of network topology. 83

Figure 5.5 Choices of W in Random 40 84

Figure 5.6 Choices of window size in Random 40 85

Figure 6.1 Delay constraints in different network topologies 107

Figure 6.2 Impact of varying arrival rates in network topologies 111

Figure 6.3 Impact of varying non-uniform traffic 115

Figure 6.4 Impact of Bursty traffic 117

Figure 6.5 Impact of heterogeneous delay constraint 120

Figure 6.6 CBR and DBR Stability and sensitivity to their parameters 124

Figure 6.7 Average Jain's index in Random 80..... 128

Figure 7.1 A random topology with ten subnetworks, each comprised of random 40 randomly connected nodes..... 142

Figure 7.2 A random topology with seven heterogeneous subnetworks 143

Figure 7.3 A lattice topology with nine subnetworks each comprised of 20 randomly connected nodes..... 144

Figure 7.4 Flow blocking probability in heterogeneous network 151

Figure 7.5 Impact of network topology..... 153

Figure 7.6 Impact of varying non-uniform traffic in heterogeneous topology ... 155

Figure 7.7 Impact of bursty traffic 159

Figure 7.8 Impact of heterogeneous traffic in lattice topology..... 160

Figure 7.9 Carried load fluctuation 162

Figure 7.10 Fluctuation in utilization of two links in heterogeneous topology .. 164

Figure 7.11 Response to rapid arrival variation in lattice topology..... 166

Figure 7.12 Average Jain's index in heterogeneous topology..... 168

Figure 7.13 Window size (connection requests) in heterogeneous topology..... 169

List of Principal Abbreviations

AS	Autonomous System
ATM	Asynchronous Transfer Mode
BBR	Bandwidth Based Routing
BFS	Breadth First Search
BSP	Bandwidth Inversion Shortest Path
CBR	Constraint Based Routing
DAR	Dynamic Alternative Routing
DBR	Delay Based Routing
DCR	Delay-Constrained Routing
DCCR	Delay Cost Constrained Routing
DFS	Depth First Search
DiffServ	Differentiated Services
DRA	Distributed Routing Algorithm
EBSP	Enhanced Bandwidth Inversion Shortest Path
FTP	File Transfer Protocol
GUI	Graphical User Interface
HBBR	Hierarchical Bandwidth Based Routing
HCBR	Hierarchical Credit Based Routing
IETF	Internet Engineering Task Force

IntServ	Integrated Services
IP	Internet Protocol
IS-IS	Intermediate System to Intermediate System
ISP	Internet Service Provider
ITU-T	International Telecommunication Union - Telecommunications
LDP	Least Delay Path
LSA	Link State Advertisement
MCP	Multi Constrained Problem
MPLS	Multi Protocol Label Switching
NED	Network Description Language
OSPF	Open Shortest Path First
PNNI	Private Network to Network Interface
PSR	Proportional Sticky Routing
QoS	Quality of Service
RIP	Routing Information Protocol
RVSP	Resource Reservation Protocol
SDP	Shortest Distance Path
SLA	Service Level Agreement
SP	Shortest Path
SRA	Source Routing Algorithm

SWP	Shortest Widest Path
TCP	Transmission Control Protocol
TE	Traffic Engineering
UDP	User Datagram Protocol
VoIP	Voice over IP
WSP	Widest Shortest Path

List of Principal Symbols

b	The requested bandwidth
bw(ℓ)	The available bandwidth for a link l
C	The maximum capacity of the links in the network
D	The delay constraint
d (ℓ)	The delay for a link l
G (V, E)	A network *G* with nodes *V* and links *E*
h	The average path length (number of hops) in the network
L	The number of links in the network
L_d	The offered load in the network
N	The number of nodes in the network
O (*N*)	The complexity of order *N*
P	Path
$\boldsymbol{P}^{alt}$	The Alternative path with maximum credit
$\boldsymbol{P}^{\min}$	The Minimum Hop path with maximum credit
$\boldsymbol{P}^{\min}$***.credits***	The credit of the minimum hop path
$\boldsymbol{P}^{alt}$***. credits***	The credit of the alternative path

$\boldsymbol{P}$**avg**	The average residual bandwidth for a path p
$\boldsymbol{R}$	The Set of Candidate Path
$\boldsymbol{R^{min}}$	the Set of Minimum Hop Candidate Path
$\boldsymbol{R^{alt}}$	the Set of Alternative Candidate Path
$\boldsymbol{W}$	Sliding window Length

Chapter 1

Introduction

1.1 Background

The existing Internet network has been built up as an enormous packet-switched network. The use of the Internet has grown rapidly due to the emergence of the World Wide Web and also real-time traffic on the Internet for the applications with new characteristics and new requirements has increased significantly. Despite the fact that the Internet witnesses tremendous success, its service model and architecture have remained the same. Routing in the Internet is still a datagram routing network providing a "best-effort" paradigm in which all packets are served indistinguishably. The best-effort service provides the simplest form of service that the network can offer, so it does not offer any guarantee to applications traffic. The best-effort service works well with data applications, such as telnet, e-mail and file transfer, which can tolerate large variation delay and packet losses. This is not appropriate for real-time applications which are less tolerant for large delay variation and packet losses caused by network congestion. For these reasons new architectures and technologies are needed for the internet to

support resource assurance, a variety of communication services, resource allocation and the evolving into a multi-service communication network.

Quality of Service (QoS) is a new mechanism proposed to manage network resources in order to deliver reliable and expected performance over the Internet in terms of delay, loss probability, throughput, and availability based on application requirements. QoS provides either a guarantee for the required service for real time applications or better services for these applications to satisfy the minimum service required.

QoS routing is a key element in any QoS architecture. The main objective of a QoS routing algorithm is to find a path that satisfies an application's requirements. QoS routing schemes take into consideration the state of link state information and based on this finds a feasible path that satisfies the QoS requirements of each connection request. The QoS routing schemes also need to minimize routing and computational overhead and at the same time maximize the utilization of the network resources.

1.2 Motivation

Routing in communication networks is the process of transporting information or data from a source node to the destination node across a network. A QoS routing protocol is required to determine, distribute and keep up to date the set of dynamic changes in network state information (e.g. links weights). The routing protocols have the ability to notify each router a view of the network state using a link state update policy. This information however changes rapidly compared to information exchanged about network topology, as in best-effort routing. Based on this information, a QoS routing algorithm can determine the best path for a given connection request to satisfy its QoS requirements and at the same time utilize the network resources efficiently. However, it is not practical to suppose that each network node has accurate link state information of all links in the network at all times, since to keep absolutely up to date information would require a prohibitively extensive exchange of link state update information between network nodes for all links and this will consume an unacceptable amount of network resources. Most of the existing link state routing algorithms consider a trade off between the link state update overhead and the accuracy of network resources [1]. However, due to the extremely dynamic nature of the internet traffic and using large update intervals to reduce routing overhead, stale/outdated information will occur. Furthermore, using large update intervals of global state

information may lead to route flapping[1]. Such behaviour occurs when the utilization on a link is low and the out-of-date information causes all nodes to route traffic along this link, resulting in rapid utilization of this link. Likewise, with high utilization, all source nodes avoid using this link and its utilization decreases. Such oscillatory behaviour results in poor route selection, instability and an overall degradation of network performance.

The above problems with existing QoS routing become a serious issues as the sized of a network increases and therefore the scalability of QoS routing algorithms becomes a major challenge. The inherent scalability problems of global QoS routing algorithms therefore motivates us to study and develop new QoS routing methods for enhancing the scalability of QoS routing algorithms.

1.3 Thesis Aims and Objectives

In this thesis, the major aim will be how to enhance the scalability of QoS routing. The research is focused on localized QoS routing algorithms with the set of sub aims listed below.

- To develop scalable, localized QoS routing algorithms that select a path based on available resources.
- To develop a hierarchical localized QoS routing algorithm in two levels of a hierarchical network.

These aims are to be achieved through the following objectives:

- To study several approaches of global and local QoS routing and related scalability problems of global routing.
- To develop localized QoS routing algorithms that provide a connection with guaranteed QoS requirements, low message overhead and efficient resource utilization.
- To develop an efficient and scalable localised QoS routing algorithm using the bandwidth metric as the path selection criteria.
- To develop novel localized QoS routing that uses delay as the QoS metric.
- To develop a scalable and efficient hierarchical scheme based on localized information that does not require global information for each hierarchical level.

- To develop a simulation environment that can be used to assess the performance of the proposed localized algorithms against other localized and global QoS algorithms.

1.4 Thesis contributions

This thesis makes several contributions as follows:

- We have presented a bandwidth based localised algorithm for QoS routing that performs routing using only flow statistics collected locally about average residual bandwidth. We have compared its performance against existing global and localized QoS routing algorithms and demonstrate through extensive simulations that the algorithm performs well in comparison to these.
- We have developed delay-based routing which is a simple localized QoS routing algorithm that relies on average delay on the path in order to take routing decisions.
- We study other QoS routing schemes; such as localized credit-based routing (CBR) proposed in [2] and this scheme has been modified to use delay instead of bandwidth.
- The two localized delay algorithms and the commonly used global shortest path algorithm (Dijkstra's) and shortest-widest-path (SWP) are compared under different delay constraints and network topologies. We demonstrated

through simulation that the localized schemes outperform Dijkstra and SWP schemes in all network topologies, unless unrealistically small update intervals of link state are used for the global schemes.

- We have proposed a scalable and efficient hierarchical routing scheme based on localized information that does not require global information for each hierarchical level in two levels of a hierarchal network.
- We have developed the localized Credit Based Routing (CBR) so this is applicable to hierarchical networks.
- We have developed a two level hierarchical global QoS routing algorithm. The algorithm uses Widest Path (Dijkstra) in the lower level and the widest shortest path (WSP) algorithm for the backbone We demonstrate through simulation that our scheme performs better than the modified CBR and outperforms algorithms that use the basics of schemes that currently operate on the internet, under different ranges of workloads and system parameters, even for a small update interval of link state.

1.5 Thesis outline

The structure of the thesis is organised as follows.

Chapter 2 introduces the QoS routing protocols in the Internet describing Internet QoS mechanisms, architectures and IP-QoS service classes. A comprehensive description on QoS routing notation and metrics, how state information is maintained and update frequency is also provided. We give an explanation about the QoS-routing problem and discuss the advantages and limitations of different routing strategies.

Chapter 3 describes the time complexity of QoS routing and gives an overview of unicast QoS routing algorithms based on shortest path algorithms, global routing algorithms and localised routing algorithms.

We discuss in **Chapter 4** the simulation model that is used to assess the performance of the algorithms developed in Chapter 5, 6, 7. We also describe the simulator design and how it was validated. Different types of network topologies, parameter settings and the performance metrics used in the performance evaluation are also described.

In **chapter 5** we introduce and describe a new localized bandwidth based QoS routing algorithm (BBR). This is followed by an extensive simulation evaluation of the proposed algorithm against the other localized routing CBR and the global QoS routing WSP.

Chapter 6 proposes a novel delay based QoS routing algorithm (DBR) and develops the localized Credit Based Routing (CBR) so this can use delay instead of bandwidth. A comprehensive description of both algorithms is also provided. We conclude **Chapter 6** with simulation evaluation and results of the proposed algorithm against the global shortest path algorithm (Dijkstra's) and shortest-widest-path (SWP).

In **Chapter 7** we introduce a localized QoS routing mechanism in hierarchical networks and develop a two-level hierarchical localized QoS routing algorithm (HBBR). We have also developed the localized Credit Based Routing (CBR) so this is applicable to hierarchical networks. In order to be able to measure the performance of our hierarchical routing scheme, two-level hierarchical networks have been generated for use in the simulations. Finally, we conclude **Chapter 7** with results from the simulations.

Chapter 8 concludes the thesis and points out possible future directions for the work.

Chapter 2

QoS Routing

2.1 Introduction

The internet is a collection of networks interconnected to each other through gateway routers. Routing can be defined as the process of delivering information from a source to a destination through intermediate routers. Routing consists of two main operations: finding a path between any source and a destination pair using routing algorithms such as the Dijkstra and Bellman-Ford's shortest path algorithms [3] [4]; whereas routing protocols such as Open Shortest Path First (OSPF) [5] and Routing Internet Protocol (RIP) [6] are responsible for delivering the data (packets) once the path is selected. When a router receives a packet it uses the information stored in the packet's header and the forwarding table to make a routing decision. This information is stored within routers and it is important for efficient routing. Packets sent via the Internet Protocol (IP) over the internet are treated equally, but this is not the case for applications that need different levels of service assurance. The current Internet Protocol (IP) provides a single level of service which is not sufficient for real time and multimedia

applications. Unfortunately, the ‘best-effort’ is the only class of service offered for the internet. In best-effort networks, the routing protocol focuses on the connectivity of network nodes and their links. Moreover, best-effort networks do not maintain information as each connection arrives at routers, so there is no resource reservation or admission control for each connection. Since there is no guarantee of packet delivery, the end host needs to provide reliable packet delivery. The Transmission Control Protocol (TCP) [7] is used by the end host to retransmit packets in case there is no acknowledgment of proper delivery, whereas the User Datagram Protocol (UDP) [8] is used for applications that rely solely on best-effort.

Routing protocols used in best-effort networks, such as the internet, are simple and scalable but have several drawbacks. First, they primarily use a shortest path routing technique that uses only a single minimum hop path between each source and destination, which can cause uneven distribution of traffic. They also use single routing metrics, such as hop count. They do not have admission control and do not differentiate between traffic that uses the network; thus, all connection requests are accepted to the network, leading to network congestion. Moreover, packets are sent to the network without delivery guarantee, which can lead to packet loss or drop along the shortest path.

2.2 Internet Quality of Service: Mechanism and Architecture

The modernization of the internet that has brought it into commercial use has brought about certain challenges in terms of performance and the services offered. Various real-time applications need different levels of service which are not provided by the current internet. The current internet is a datagram model and connectionless network that has imperfect resource management. In a datagram model, different routes may be used to send packets of a session to a destination, and hence may be received out of their original order. Such behaviour reflects on the quality of real-time applications such as video on demand and video conferencing. Moreover, routing in the internet does not route traffic along alternative paths when the main path is overloaded. Since the internet has become a vital part of people's daily activity, there is a need to re-develop the internet so that it is efficient enough to support real-time applications with real quality of service guaranteed.

Quality of Service (QoS) can be defined as "a set of service requirements to be met by the network while transporting a flow" [9]. It is also defined as "the collective effect of service performances which determine the degree of satisfaction of a user of the service" [10]. Many mechanisms have been proposed to ensure provision of a QoS protocol in the internet. The following sections

describe the most noted architectures, namely Integrated Services and Resource Reservation Signalling Protocol architecture (IntServ/RVSP) [11],[12], the Differentiated Services (DiffServ) architecture [13], Multi-Protocol Label Switching (MPLS) [14], Constraint Based Routing (CBR) [9], and Traffic Engineering (TE) [15].

2.2.1 Integrated Service (Intserv) [11]

The Intserv approach is a per-flow service. The requirements of a given flow can be guaranteed by reserving resources, such as bandwidth and buffers, explicitly to ensure that each flow receives its requested service. With the Intserv approach each device in the network must participate by reserving resources and isolating each flow from the other. RSVP is used by Intserv to reserve network resources and set up a flow with specific QoS for an application flow. It is also used to deliver the QoS requirements to all intermediate routers along the selected path and to record and maintain the state of each flow to provide the QoS requested [16]. According to the availability of network resources to satisfy a new connection arrival, the connection will be admitted to the network; otherwise, the connection will be rejected. RSVP requires the sender or the receiver of the admitted flow to establish a soft state to manage resources within routers. RSVP soft state is established and periodically refreshed to avoid termination of the flow using RSVP messages. However, with the unpredictable growth of the internet

and the huge growth of state information, IntServ/RVSP requires routers to manage very large numbers of flows and causes signalling overhead of applying RVSP protocol; hence, IntServ does not scale well in the internet.

2.2.2 Differentiated Services (DiffServ) [13]

Differentiated Services (DiffServ) have been developed to solve the scalability problem raised by Integrated Services. Routers in integrated services should be able to differentiate between large numbers of connection requests, which result in large overhead to maintain a connection request state table. Differentiated Services reduce the complexity in core routers by dividing traffic into aggregated numbers of forwarding classes in the edge router that is responsible for classifying packets into their proper class. The classification of the packets is done based on the service level agreement (SLA) between service providers. Differentiated Services do not require advance resource reservation setup and the classification of packets is done on the edge of the network. This makes DiffServ more scalable and flexible. The DiffServ field of the Internet Protocol (IP) header is marked with the type of service level of agreement (SLA) applicable; hence, when routers receive marked packets they work out how the packet can be processed. DiffServ has Per Hop Behaviour (PHB), where the best-effort treatment is called DEfault per Hop Behaviour (DE PHB) because the networks do not provide the required quality of service. Expedited Forwarding per Hop Behaviour (EF PHB) [17]

provides applications with low loss, low jitter, and low latency. Low priority traffic is provided by Assured Forwarding per Hop Behaviour (AF PHB) [18] and delivered packets may be dropped and given low priority in case of congestion.

Although DiffServ is scalable and has less signalling overhead than IntServ/RVSP, it does not provide full guarantees during congestion and does not provide end to end guarantees for real time applications

2.2.3 Multiprotocol Label Switching (MPLS) [14]

Routing decisions in IP networks today are solely based on destination address, which is inefficient when applying routing policy. Multiprotocol Label Switching (MPLS) [19] is a new forwarding scheme where packets are forwarded based on an attached label. MPLS has a header between the network layer and data link layer in the IP header [20]. To set up a path in an MPLS domain a signalling Label Switching Protocol (LSP) is used, and if the flow is admitted then packets are assigned an MPLS label which determines the path that will be used in that domain. Label switch routers (LSRs) are responsible for swapping labels in MPLS headers to forward packets on the determined path to their destinations. MPLS provides fast packet forwarding as the added labels can be switched more efficiently by LSR routers, which is a requirement for large networks. Furthermore, MPLS label switched paths can by used to deploy a virtual private network (VPN) [21]. MPLS can set up an explicit path using a label switched in

the MPLS domain, since the path used by the packet is specified by a single LSR router and should not be carried in the packet header.

2.2.4 Traffic Engineering (TE) and Constraint Based Routing (CBR) [22]

Traffic Engineering is the process of managing traffic flows on an IP network to avoid network congestion. Such congestion is caused by uneven traffic distribution resulting from using shortest path protocols such as OSPF, RIP and IS-IS. Uneven distribution of traffic causes overload in some network links, while others stay underutilized. Traffic Engineering provides an advanced mechanism to allocate traffic in the internet. This requires source nodes in the network to take into consideration available bandwidth in the network before computing paths.

Constraint Based Routing (CBR) is a set of protocols and algorithms that facilitate a source node to compute a feasible path to a destination node, taking into account multiple constraints, to increase the utilization of the network [22]. CBR constraints can be administrative costs or application QoS requirements for applications such as bandwidth, delay, jitter and packet loss [23]. Requirements for applications to be satisfied can be named as QoS routing. It is obvious that the most important QoS mechanism in the internet is QoS routing, which makes the traffic engineering process automatic [24].

2.3 QoS Routing

Quality of Service, abbreviated as QoS, is defined as a "routing mechanism under which paths for flows are determined based on some knowledge of resource availability in the network as well as the QoS requirements of flows", where a flow is "a packet stream from source to a destination with an associated Quality of Service (QoS)" [9]. QoS requirements need to be expressed in some assessable QoS routing metrics, such as delay, number of hops, bandwidth, cost, and jitter. The main objectives of QoS routing are as follows [9]:

1. Dynamic determination of feasible paths that satisfy the requirements of a flow.
2. Network resource optimization and improvement of overall performance by efficient distribution of the traffic in the network and maximization of its resource utilization.
3. Avoidance of congestion hotspots in the network and provision of good performance with heavy loads.

2.3.1 Notations and Metrics

A network can be represented as a directed graph *G (V, E)*, where *V* represents the set of nodes (routers) and *E* represents the set of arcs (links) that connect nodes in the network. Each link belongs to *E* from node *u* to node *v*, represented by *(u, v)*,

and has k values of its metric, one for each link, represented as k weights: $w_1(u,v), w_2(u,v), w_3(u,v), ..., w_k(u,v)$ where $w_i(u,v) > 0 \ \forall (u,v) \in E$. Path p from node s to node d in G is represented by $p = u_{s=0} \to u_1 \to u_2 \to ... \to u_{d=l}$ such that $(u_i, u_{i+1}) \in E$ for all $1 \le i \le l-1$.

QoS routing algorithms need to find the path P that satisfies QoS requirements of a flow. These requirements are measured using certain metrics which need to be selected so that the requirements can be presented as a single metric or a combination of them. The four most common types of metrics, which are also called the composition rules of the metrics, are as follows [25]:

- A QoS metric is **additive** if $w(P) = w(u_1, u_2) + w(u_2, u_3) + \ldots + w(u_{i-1}, u_i)$.
- A QoS metric is **multiplicative** if $w(P) = w(u_1, u_2).\ w(u_2, u_3).\ldots\ldots\ w(u_{i-1}, u_i)$.
- A QoS metric is **concave** if $w(P) = \min(w(u_1, u_2), w(u_2, u_3), \ldots, w(u_{i-1}, u_i))$.
- A QoS metric is **convex** if $w(P) = \max(w(u_1, u_2), w(u_2, u_3), \ldots, w(u_{i-1}, u_i))$.

Bandwidth is the most widely used metric and is a concave metric also called a link metric. Delay, delay jitter, cost and hop count are additive metrics. With non-additive metrics, pruning can be used to reduce computation overhead by logically removing the links that do not satisfy QoS requirements. Multiplicative metrics can be computed as additive metrics by replacing the link weights and constraint by their logarithms.

2.3.2 Maintenance of State Information

In a QoS routing process, finding a feasible path that satisfies flow requirements requires knowledge of the network state, which must be kept up-to-date. Routing protocols provide each node in the network with information about the network state. A routing algorithm is responsible for finding a feasible path for a new connection, based on the information gathered by the routing protocol. The efficiency of any routing depends on the way in which the network state information is collected and kept up-to-date. Information about the network state is collected based on a global, localized, or aggregated approach, as discussed below:

Local state information

With local state information, each node is responsible for keeping the latest information collected locally, usually from adjacent nodes, about its flow. This information can be bandwidth, delay, or any QoS metric. The collected statistics are then used as state information to find a feasible path for a new connection [26].

Global state information

Global state information is a combination of local state information in all nodes. It is maintained in each node by periodic exchange of link QoS state information among network nodes obtained from a global view of the information. Within the

global QoS state there are two different protocols used to maintain and collect the global state in each node [3]. The Distance Vector Protocol refers to a simple routing protocol that uses distance or hop count as its primary metric for choosing the best path. In the Distance Vector Protocol, routing tables are exchanged periodically between neighbouring nodes and it is assumed that each router knows the distance to its neighbours. Alternatively, the Link State Protocol is more complex as it updates other routers' information to determine the best path based on a specific QoS metric. Since routers need information about the availability of resources in the network in order to compute routes supporting the QoS requirements of a flow, Link State Protocols enable link state routers to update neighbouring networks with current information, rather than continually providing routing tables to detect change in the state of the routing path.

Aggregated state information

Growth in network size causes difficulties in maintaining global network state information. Thus, a hierarchical topology has been proposed, clustering the nodes into groups to form logical nodes. The logical nodes are clustered into groups to form higher level logical nodes, and so on. Nodes in each cluster store more information about nodes in the same cluster and less information about nodes in other clusters.

2.3.3 Update Frequency and Information Inaccuracy

It is important for each node in the network to be aware of each link state which is used to compute feasible paths. QoS routing can compute paths based on pre-computation [27], on-demand [28] or path-caching [29]. The source node in the network uses one of theses methods with knowledge of global state information.

The **pre-computation** scheme requires each node to compute paths for each destination periodically. Hence, each node receives a number of updates regardless of whether the path is required or not. The updated state information is then used to select a feasible path when a new connection arrives.

The **on-demand** scheme computes paths to the destination only when a new connection arrives. The scheme has the benefit of using the most recent state information to calculate a feasible path for the new connection arrival.

The choice between the two approaches requires careful consideration in terms of how much information needs to be processed and network size. The on-demand scheme is suitable for small networks while the pre-computation scheme is better for large networks. Moreover, on-demand QoS routing algorithms are preferable when arrival of connections are infrequent, while pre-computation QoS routing algorithms perform well when connection requests are more frequent. Finally, the on-demand computation scheme is less complex as it only calculates one single feasible path, whereas the pre-computation scheme reduces the path setup time

[30]. A hybrid approach, path-caching, is proposed to reduce the computational cost by using previously computed paths.

While the performance of QoS routing algorithms rely on the maintenance of global state information, a trade off between the accuracy and density of state information should receive considerable attention [1]. It is not applicable to provide network nodes with up to date changes in the links at all times, however, as rapid global state information updates for any change in each link state would consume a huge amount of network resources. To reduce the overhead of rapid link state updates, the rate of updates needs to be reduced. In the widely used protocol OSPF, it is recommended that the link state be updated once every 30 minutes. Consequently, not all link state changes will be advertised [31]. Such a large time interval leads to stale link state information, which can affect QoS routing as follows [32]:

- A QoS routing algorithm may not find a feasible path, even though one exists.
- A path may be rejected in the setup process because of false information about available link resources.
- A QoS routing algorithm may select a non-optimal path.

In QoS routing, many link state update policies have been proposed, such as periodic, threshold-based, and class-based update policies [33], [34]. The periodic policy advertises the link state information throughout the network periodically,

thus it is easy to implement as it is not tied to traffic changes. In the threshold based policy the update triggers the relative difference between the available value known by the network and the actual current value of a specific parameter exceeds the threshold. The class based policy uses two classes to divide the QoS parameter. An update of link state is advertised when the actual current value of the QoS parameter changes from lower class to upper class, or vice versa.

2.3.4 QoS Routing Problems

Routing problems involve finding a path that satisfies a set of QoS constraints between source and destination nodes. Routing problems are generally categorised into two main classes: the unicast routing problem and the multicast routing problem. In this thesis the main focus will be on the unicast routing problem.

2.3.4.1 The Unicast Routing Problem

Connection QoS requirements can be defined as a set of constraints, classified as link constraints and path constraints. A link constraint can be defined as a limit of maximum or minimum use of link resources, and a path constraint can be defined as an end-to-end QoS limit on a path. The routing problem is classified by the number of metrics used; when a connection is satisfied by a single QoS requirement, a single constraint routing problem arises. However, multiple

constraint routing problems exist when a connection specifies multiple QoS requirements.

Single constraint routing problems

Single constraint routing problems are either constraint or optimization problems. They can be divided into four types:

- *Path optimization problem*: finds a path that has minimum end-to-end cost, such as least-cost routing. This problem can be solved by directly using the standard shortest path algorithm.
- *Path constraint problem*: finds a path that has least additive end-to-end QoS metric, such as end-to-end delay. This problem also can be solved by directly using the standard shortest path algorithm
- *Link optimization problem*: finds a path with the maximum concave metric, such as largest bottleneck bandwidth. This problem can be solved using a modified version of Dijkstra's algorithm or Bellman-Ford's algorithm.
- *Link constraint problem*: this problem can be reduced to the link optimization problem, such as finding a path whose link bandwidths are equal or higher than a specified bandwidth requirement. Pruning links that have less than the required bandwidth and then finding the shortest path in the pruned topology is another approach.

Multiple constraint routing problem

The multiple constraint routing problems can be formulated from the aforementioned four basic problems. Routing problems that are composed of one additive and one non-additive metric can easily be solved under polynomial time using standard shortest path algorithms. However, if the routing problem is composed of two or more additive metrics then it is known to be NP-complete [25] [35], which implies that there is no known efficient algorithm and that one has to rely on heuristics and sub-optimal solutions. Table 2.1 lists some examples of the basic and composite problems and their complexity.

2.3.4.2 The Multicast Routing Problem

The main difference between multicast routing and multiple unicast routing to several destinations is that multicast routing is best captured by the host group model. A host group is a set of network entities sharing a common identifying multicast address, all receiving any data packets addressed to the multicast address by senders (sources) that may or may not be members of the same group and that have no knowledge of the group's membership [36].

	Problem	Example	Description	Comp lexity
Basic routing problems	link-constrained	Bandwidth- constrained	Find a path whose bottleneck bandwidth is above a given value	polynomial
	link-optimization	Bandwidth- optimization	Find the path with maximum bottleneck bandwidth	
	path-constrained	delay- constrained	Find a path with bounded delay	
	path-optimization	cost- optimization	Find a path whose total cost is minimized	
Composite routing problem	link-constrained path-optimization	delay- optimization bandwidth- constrained	Find the least-delay path with the required bandwidth	polynomial
	path-constrained link-optimization	delay- constrained Bandwidth- optimization	Find the path with maximum bottleneck bandwidth and bounded delay	
	multi-path constrained	delay- constrained cost- constrained	Find a path whose cost and delay are less than some given values	NP- complete
	path-constrained path-optimization	delay- constrained cost- optimization	Find the least-cost path with bounded delay	

Table 2.1: Some QoS unicast routing problems.

2.3.4.3 NP-Completeness

A multiple constraint routing problem involving two or more additive metrics has been shown to be NP-Complete [25, 35, 37]. The metrics were assumed to take real values and be independent. This problem has been investigated and many heuristics have been proposed to solve the NP-complete problem in polynomial time [38] [39] [40] [41]. The problem can be avoided if one of the metrics refers to bandwidth as the QoS [40].

2.3.5 QoS Routing Strategies

QoS routing algorithms can be classified into three major classes based on the way in which global state information is collected and where the path computation is carried out [42].

2.3.5.1 Source Routing

In source routing, feasible paths are completely computed at the source node. Therefore, each node needs to maintain complete global state information of the network, such as network topology and the state of each link in the network. A setup message is sent by the source node along the computed path to inform each intermediate node about a connection request requirement until it reaches the destination node. However, if no feasible path is found a source node may reject the connection or negotiate for fewer requirements. The global state information that is exchanged among network nodes in the source routing approach is used by either link state protocol [5] or distance vector protocol [43]. Since the feasible paths are computed in a centralized fashion for each individual connection, source routing is simple, flexible, loop free and easy to implement. Each source node can easily use more than one algorithm by choice in the same network since the paths are computed locally. However, source routing greatly relies on the precision of the maintenance of global state information, which involves a very frequent

exchange of complete information in order to keep it up-to-date [1] [44]. Furthermore, the computational overhead in the source routing approach is very high since it is done on one single node. Although a scalability problem occurs in source routing, a lot of QoS routing is based on this approach [42] [45] [46]. The approach allows the application of different QoS schemes, such as partitioning of large networks to reduce computational overhead [47], end to end QoS requirements partitioning to reduce routing cost [48], and traffic engineering [49].

2.3.5.2 Distributed Routing

In distributed routing, feasible path computations are distributed among the intermediate nodes between source and destination nodes. Most distributed routing algorithms [50] [51] require each node to maintain global state information and based on this routing decision is determined on a hop-by-hop basis. A routing table that stores the next hops for all destinations is computed periodically at each node and this makes the communication overhead unusually high for large networks. However, distributed routing algorithms have less setup time and are more scalable compared to source routing algorithms. Another approach for distributed routing is called Flooding-based [52] [53] QoS routing, which does not require link state information and complex path computation as it has the ability to search and select the best path based on a number of flooded control messages from the source node. Distributed routing algorithms may suffer

from routing loops when the network state is imprecise, whereas algorithms that do not require global state information require the sending of more control messages, which becomes a problem for large scale networks.

2.3.5.3 Hierarchical Routing

Hierarchical routing has been proposed to solve the scalability problem in large networks [54] [55] [56]. In hierarchical routing, nodes are clustered into groups to form a logical node. The logical nodes are further clustered into higher level logical nodes, creating a hierarchy in the form of a multi level topology. Each node is required to maintain aggregated network state information about the other clusters and detailed state information about nodes in its own cluster. Hierarchical routing algorithms use source routing algorithms to compute feasible paths as connection requests arrive. They also use distributed routing algorithms through the distribution of path computation over many nodes, so they have the advantages of both strategies. The size of the aggregated information in hierarchical algorithms is logarithmic to the size of the whole state information; hence, they reduce the computational effort and the exchange of network state overhead, and as a result they perform well with large scale networks [57]. In contrast, they suffer from imprecise state information produced by the aggregation, which increases as the number of aggregated levels increase [58].

2.3.6 Routing in the Internet

Networks can be categorised into two major categories in terms of routing paradigms they are using: flow-based networks (such as ATM and MPLS networks) and hop-by-hop networks. The flow-based networks do routing and traffic engineering based on connections (flows) [104]. In contrast, the hop-by-hop routing concept forms the basis of today's Internet, this due to the fact that it is simple, reliable, and has a widespread deployment. In fact, the most universally used protocols, for intra-domain and inter-domain, are essentially hop-by-hop routing. The common interior gateway protocol (IGP) for intra domain routing is the Routing Information Protocol (RIP) [6] which broadcast to its immediate neighbours information about all destinations known to it along with their corresponding shortest distances. The other intra-domain routing protocols is called link-state routing, represented by protocols such as Open Shortest-Path First (OSPF) [5] and Intermediate System to Intermediate System (IS-IS) [105]. In these protocols each node floods the entire autonomous system (AS) with information about network state information. For inter-domain routing the Border Gateway Protocol (BGP) [106] is used to exchange network reachability information to allow inter-AS communication.

Chapter 3

QoS Routing Algorithms

3.1 Introduction

An important issue for communication networks is how to route traffic utilizing network resources efficiently. Successful deployment of QoS routing can enable the finding of a feasible path that has adequate resources to satisfy a set of QoS requirements of a connection. QoS routing needs to carry out two main tasks in order to satisfy connection requirements: path computation and collection of state information. QoS routing protocols are responsible for collecting and maintaining state information and keeping it up to date. Based on methods for maintaining state information and the computation of feasible paths, there are three routing strategies: source routing, distributed routing, and hierarchical routing. A survey of different QoS routing algorithms can be found in [23] [59] [60]. However, there are some problems associated with QoS routing algorithms, such as complexity, optimality and scalability [10]. For the purpose of this thesis we will categorize QoS routing into global QoS routing schemes and local QoS routing

schemes. In this thesis we focus on localized QoS routing algorithms and global QoS routing algorithms are used for comparison purposes.

3.2 Time Complexity

It should be possible to scale QoS routing algorithms to large networks as this will introduce more complexity and more overhead so that feasible paths can be found. Time complexity reflects the dominant factors in the number of steps an algorithm takes to solve a problem. So the time complexity of a QoS routing algorithm is related to the number of operations needed to satisfy QoS requirements and their composition rules. In QoS source routing algorithms, time complexity is a major performance criterion, since all computational steps to find a feasible path are

Routing Algorithm	**State-Information**	**Time Complexity**
WSP	Global	O ($N \log N+L$)
SWP	Global	O ($N \log N+L$)
SDP	Global	O ($N \log N+L$)
Dijkstra	Global	O ($\| N \|^2$)
PSR	Local	O (R)
CBR	Local	O (R)
N = number of nodes, L = number of links , R = number of candidate paths		

Table 3.1: Time complexity of QoS routing algorithms [25][80] [107] .

carried out in the source [61]. Table 3.1 shows the time complexity of the main source global routing algorithms and localized routing algorithms.

3.3 Shortest Path Algorithms

The goal of shortest path routing algorithms is to find the shortest path between a given source and destination so that the cost of links used on the path is kept to a minimum. The problem of shortest paths can be solved by two well-known algorithms: the Dijkstra algorithm [62] and the Bellman-Ford algorithm [63]. The Dijkstra algorithm is capable of computing the shortest path from a given source to all destinations in the network. However, the Dijkstra algorithm is not valid for negative weight links in a network, unlike the Bellman-Ford algorithm in which the source node needs to know the cost of the shortest path to all nodes before the destination. An important feature of the Bellman-Ford algorithm is that it can be used as a distributed algorithm such as the RIP [43].

The Bellman-Ford algorithm and Dijkstra's algorithm can find shortest paths using any single additive metric such as cost, hop count, delay and distance. However, in many cases there is a need to find a path between a given source and a destination node with the maximum capacity. This is done by taking the minimum of the link capacities (the residual bandwidth) to obtain the capacity of the path [64]. It is also applicable to use the Dijkstra algorithm to find what is called the *K*-

shortest path; the algorithm can return all feasible paths between a given source and destination ordered in length [65].

3.4 Global Routing Algorithms

Many QoS routing algorithms require exchange of link state information among network nodes in order to get knowledge about the global view of network QoS state information. Based on the collected information and the current view of the state information, a source node finds a feasible path to allow flow to the destination node. Such global routing algorithms have different path selection methods and differ in how they exchange global state information. Global routing algorithms have to trade-off between the resource usage and load balancing, as well as between link state update frequencies and the accuracy of the network state. More information about global routing algorithms and network conditions can be found in [66].

The widest shortest path algorithm (WSP) [45]

The widest shortest algorithm chooses the shortest feasible path with minimum hop count among paths that satisfies the bandwidth constraints. If there is more than one path with the same hop count then the path with the maximum available bandwidth is selected. The widest path is only chosen if there is more than one path with the same length. The WSP algorithm minimizes the usage of network resources by preferring the shortest path to the destination. Pruning is used to

eliminate links that do not satisfy flow requirements, and then WSP is used on the pruned topology [67]. It should be noted that the difference in the performance between WSP with and without pruning is remarkable. The WSP algorithm has been studied extensively in the literature and many variations can be derived by choosing different cost functions of the shortest path [68] [69] [70].

The shortest widest path algorithm (SWP) [25]

The shortest widest algorithm finds the widest feasible path with maximum available bandwidth. If there is more than one path with the same width then the shortest path is chosen. In shortest widest path, Dijkstra's algorithm is applied twice in order to find the most feasible path. The second metric (hop count or delay) is only used if there is more then one path with equal bottleneck. The SWP algorithm prefers the widest path so that it can distribute load efficiently in the network and avoid congestion on short paths.

The shortest distance path algorithm (SDP) [64] [71] (bandwidth-inversion shortest path *bsp*)

The shortest distance algorithm selects the path with the shortest distance. The link's distance is the inverse of the available bandwidth of that link. The overall distance of a path is the sum of distances over all the links along the path according to the distance function:

$disp(p) = \sum_{i \in p} \frac{1}{w(i)}$, where *w(i)* is the available bandwidth of link *i*.

The SDP algorithm prefers the least loaded paths and takes into consideration hop counts [64]. Furthermore, the SDP algorithm can be modified to solve the shortest cost using the following function:

$disp(p,n) = \sum_{i \in p} \frac{1}{w(i)^n}$, where *w(i)* is the available bandwidth of link *i*.

Changing *n* in the SDP algorithm results in a wide range selecting between the shortest path (*n*=0) and widest path ($n \to \infty$) [71].

Enhanced bandwidth-inversion shortest path algorithm (EBSP)[72]

The EBSP algorithm is proposed to enhance the SDP algorithm by adding a penalty to the function weight of the SDP algorithm. As the number of hop counts along the path is increased the penalty is increased, which prevents the paths from being long.

$disp(p) = \sum_{j=1}^{k} \frac{2^{j-1}}{w(i_j)}$, where *w(i)* is the available bandwidth of link *i*.

QoS extensions to OSPF (QOSPF) [45]

The QOSPF algorithm is proposed to improve performance traffic with minimum impact on the existing OSPF protocol and its structure. In this algorithm hop count is computed while performing the path selection algorithm, but the bandwidth is advertised to nodes in the network. Network topology and the bandwidth link state information database need to be maintained at each node in the network. Hop count and bandwidth are used in the path selection of the QOSPF algorithm, in which three types of widest shortest path (WSP) are implemented with different computation methods:

- **Pre-computation Bellman-Ford** is used to calculate the maximum bandwidth paths (optimal paths) with minimum hop counts for all possible destinations.
- **On-demand Dijkstra** is used to calculate a feasible path to the destination based on the required bandwidth; so it does not need a routing table as it uses the current link state information. The links with insufficient bandwidth are pruned from the network and then the Dijkstra minimum hop is run on the pruned network.
- **Pre-computation Dijkstra** minimum hop is used to find paths to all possible destinations with a set of quantized bandwidth values; that is, the bandwidth capacity is divided into classes and the algorithm finds the minimum hop count for a given bandwidth.

3.5 Localized Routing Algorithms

The localized quality of service routing schemes [73] [74] has recently been proposed as an alternative to the global routing scheme, where routing decisions are based on the whole network state. In the global network scheme each node periodically exchanges information with the other nodes to obtain a global view of the network QoS state. However, in localized QoS routing each source node infers the network QoS state based on flow statistics collected locally, and performs flow routing using this localized view of the network QoS state. Source nodes must maintain a predetermined set of candidate paths to each possible destination to send the flow along these paths. These should be selected carefully using one of the existing methods or finding out which method gives the best results. Localized QoS routing also avoids drawbacks of selecting best-path routing. Instead, the localized approach considers the proportionate adjustments for selecting a path at a network node by dynamically judging its quality and traffic flow, based on local information. Furthermore, the localized routing approach increases network performance through minimal communication overheads, no processing of overheads at core routers, and easy deployment of information.

In localized QoS routing each source node is required to first determine a set of candidate paths to each possible destination. Candidate path selection is an

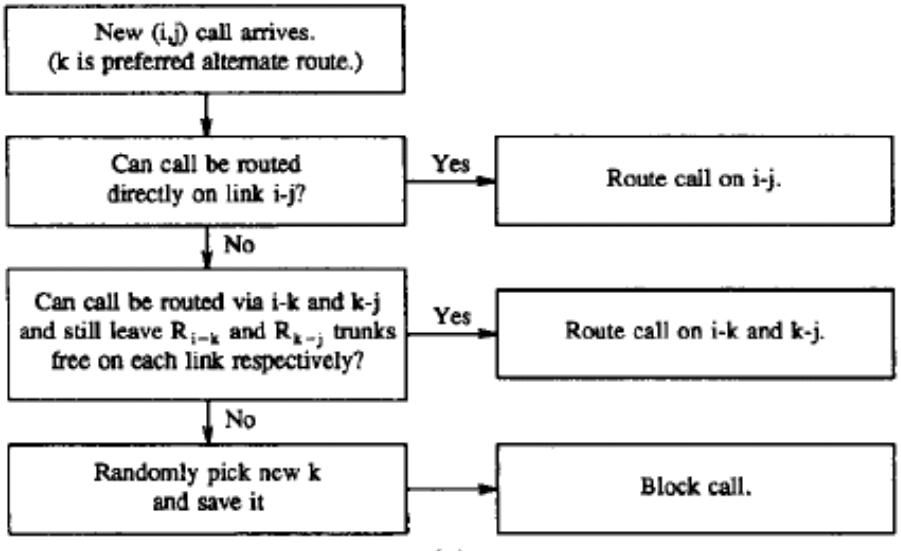


Figure 3.1 Flow diagram for DAR algorithm [77]

important factor in localized QoS routing and has a remarkable impact on its performance. More information about candidate path selection methods and their performance can be found in [75] [76]. There are various localized QoS routing approaches, some of which are discussed hereunder.

Dynamic alternative routing algorithm (DAR) [77]

The original idea of using local information in routing has been used in networks that support telephone communication [78] [79]. The methods used in telephone networks route a flow according to the feedback received from the previous accepted or rejected flows. In the DAR scheme the source node tries to route the call through a direct one-link path to the destination. If the call is not routed a preferred two-link path is then chosen to route the call. If the call cannot be routed

along the preferred two-link path then the call is blocked and another two-link path is selected randomly from all two-link paths to be the new preferred two-link path. A flow to the destination is always routed along a preferred path, which is remembered by the source node for each destination.

Learning automata based routing [79]

In this scheme, when a flow arrives it is routed along path *r* according to a probability distribution p_r. This probability is updated using feedback upon flow acceptance or rejection. The scheme uses a simple procedure of rewarding the path upon flow acceptance and penalizing it upon flow rejection. The updating equation for path *i* is chosen at time *n* if the flow accepted is:

$$p_i(n+1) = p_i(n) + a(1 - p_i(n))$$

$$p_j(n+1) = (1-a)\,p_j(n) \quad j \neq i$$

If the flow is rejected the updating equation is:

$$p_i(n+1) = (1-\varepsilon)\,p_j(n)$$

$$p_j(n+1) = \frac{\varepsilon}{r-1} + (1-\varepsilon)\,p_j(n) \quad j \neq i$$

Where a and ε are adjustable parameters, $0 < a < 1$, $0 < \varepsilon < 1$ with ε small compared with a, and a is itself usually small, so that the updating is gradual. This scheme does not take the path length or the selection of candidate paths into consideration.

```
PROCEDURE PSR-ROUTE()
  Select an eligible path r = wrrps(R^elg)
  Increment flow counter, n_r = n_r + 1
  If failed to setup connection along r
    Decrement failure counter, f_r = f_r − 1
    If failures reached limit, f_r == 0
      Remove r from eligible set, R^elg = R^elg − r
      If eligible set is empty, R^elg == ∅
        Reset eligible set, R^elg = R
        For each path r ∈ R
          Reset failure counter, f_r = γ_r
END PROCEDURE
```

(a) Proportional routing

```
PROCEDURE PSR-PROPO-COMPU()
  For each path r ∈ R
    Compute blocking probability, b_r = η γ_r / n_r
    Assign a proportion, α_r = n_r / Σ_{r̃∈R} n_r̃
  Set target blocking probability, b* = min_{r∈R^min} b_r
  For each alternative path r' ∈ R^alt
    If blocking probability high, b_r' >= b*
      Decrement failure limit, γ_r' = γ_r' − 1
    If blocking probability low, b_r' < ψ b*
      Increment failure limit, γ_r' = γ_r' + 1
END PROCEDURE
```

(b) computation of proportions

Figure 3.2 Pseudo-code for PSR(taken from [74]).

Proportional Sticky Routing (PSR) algorithm [74]

The Localized Proportional Sticky Routing (PSR) algorithm was the first localized QoS routing scheme. With this algorithm each source node needs to maintain a set of candidate paths *R*. A path is based on flow blocking probability and the load is proportionally distributed to the destination among the predefined paths. In PSR there are minimum hop (minhop) paths R^{min} and alternative paths R^{alt}, where $R = R^{min} \cup R^{alt}$. The PSR algorithm operates in two stages: proportional flow routing and computation of flow proportions. PSR proceeds in cycles of variable length, which form an observation period.

Incoming flows are routed during each cycle along a path *r* and selected based on a flow proportion from a set of eligible paths R^{alt}. Initially, all candidate paths are eligible paths and each of them are associated with an adjustable variable called the maximum permissible flow blocking parameter γ_r, which gives the maximum

number of flows before the path becomes ineligible. For each minhop path, γ_r is set to $\hat{y}$, which is a configurable parameter, whereas the alternative path γ_r is dynamically adjusted between 1 and $\hat{y}$. When all paths become ineligible a cycle is completed and a new one is started after all parameters are reset. An eligible path is selected to route the flow based on its flow proportion. At the end of the observation period, a new flow proportion α_r is computed for each path in the candidate path set, based on its observed blocking probability b_r. After each observation period the minhop path flow proportions are adjusted to equalize their blocking probability ($\alpha_r.b_r$). For the alternative paths, the minimum blocking probability among the minhop paths b* is used to control their flow proportion. That is, for each $r \in R^{alt}$, if $b_r<\psi b^*$, γ_r=min (γ_r+1, $\hat{y}$). If $b_r> b^*$, γ_r=max (γ_r-1, 1), where ψ is a configurable parameter to limit the 'knock-on' effect under system overloads. Note that $\gamma_r \geq 1$ ensures that some flows are routed along alternative paths to measure their quality.

Localized Credit Based QoS Routing (CBR) [80]

The Credit Based Routing (CBR) algorithm uses a simple routing procedure to route flows across a network. The CBR scheme uses a crediting scheme for each path in a candidate path set that rewards a path upon flow acceptance and penalizes it upon flow rejection. The path selection relies on the path's credits: the path with the largest credits among the candidate paths is chosen to send the flow.

The CBR algorithm keeps updating each path's credits upon flow acceptance and rejection and does not compute a flow proportion. It also keeps monitoring the flow blocking probabilities for each path and adds this information to the crediting scheme for use in future path selection.

A set of candidate paths *R* between each source and destination is required in the CBR algorithm. Like PSR, CBR predetermines a minhop path set $R^{\min}$ and an alternative path set R^{alt}, where $R = R^{\min} \cup R^{alt}$. CBR selects the largest credit path *P*.credits in each set, minhop path set $R^{\min}$ and alternative path set R^{alt} upon flow arrival. The flow is routed along the minhop path that has the largest credit $P^{\min}$ which is larger than the alternative path that has the largest credit P^{alt} ; otherwise the flow is routed along an alternative path if (1) is satisfied.

$P^{\min}.credits >= \Phi \times P^{alt}.credits$, where $\Phi \leq 1$ (1)

Φ, is a system parameter that controls the usage of alternative paths. The CBR uses blocking probability in crediting schemes to improve the algorithm's performance, as a path with low blocking probability will gain more credits. Path credits are increased and decreased upon flow acceptance and rejection respectively using blocking probability of the path. However, the CBR uses a MAX_CREDITS parameter to determine the maximum attainable credits for each path using (2).

$0 \leq$ credits $\leq$ MAX_CREDITS (2)

The CBR algorithm records rejection and acceptance for each path and uses a sliding window for a predetermined period of M connection requests. It uses 1 for flow acceptance and 0 for flow rejection, dividing the number of 0's by M to estimate each path's blocking probability for a period of M connection requests.

Cognitive Packet Networks (CPN) [108] [109]

The Cognitive Packet Network (CPN) is a network architecture that searches for a path based on user QoS requirements using adaptive techniques. Cognitive packets learn to avoid congestion in the network in order to decrease packets lost or delay. By achieving such requirements, the overall reliability of the network can be increased. Cognitive Packet Networks carry three types of packets: smart packets, dumb packets and acknowledgments. Smart packets explore routes between source and destination nodes and avoid link and node failures, congestion, and getting lost. These packets gain knowledge about routes by observations the network status and from the experience of other packets. A notable benefit of the CPN is that it does not need much storage on the routers. Reinforcement learning is used by smart packets to discover routes, and the reinforcement learning reward function incorporates the QoS requested by a particular user. Upon arrival of a smart packet at its destination node, the destination node generates an acknowledgement, which will follow the reverse route that the smart packet passed on its way to the destination node. As it

traverses successive nodes, it updates mailboxes in the CPN nodes; when it reaches the source node, it provides source routing information for the dumb packets. Dumb packets of a specific QoS class use successful routes that have been selected in this manner by the smart packets of the same class.

Ticket-Based probing algorithm [110]

This scheme is a distributed algorithm that tries to find a low cost path subject to bandwidth or delay constraints. Each source node keeps estimates of bandwidth, delay, and cost to every possible destination. To model imprecision in links states, it also keeps estimates of variations in bandwidth and delay. This information is assumed to be updated periodically using conventional link-state or distance-vector protocols. The algorithm starts by having the source node to send probe messages toward the destination to search for a low cost path that satisfies the delay (bandwidth) constraint. Each probe carries one or more so called tickets. The authors propose rules for distributing tickets based on the likelihood of finding a feasible path. Probes are forwarded along links that satisfy the delay (bandwidth) constraint. The tickets are coloured either yellow or green. Yellow tickets prefer paths with smaller delay (larger residual bandwidth) while green tickets prefer paths with smaller cost. When one or more valid probes reach the destination, this means that a feasible path has been found and the path with the least cost is selected.

On-Policy First-Visit Monte Carlo (ONMC) [111]

The ONMC scheme is a multipath distributed routing algorithm for supporting end-to-end delay or bandwidth requirements proposed to tolerate high degrees of imprecise state information. It is proposed to maximise the success probability to find a feasible path in absence of accurate information in dynamic networks. This scheme studies a delay-constraint least-cost problem. In this algorithm there are two tasks. Firstly, a suitable number of tickets (M_0) are determined by the ONMC. In [110], $M_0 = Y_0 + G_0$ where Y_0 and G_0 are the number of yellow and green tickets, respectively. These two types of tickets have different purposes. The yellow tickets are for increasing the probability of finding feasible paths while the green tickets are for increasing the probability low cost paths. M_0 is selected from some finite set in a sequential decision-making process in the presence of state uncertainty with the objective of maximizing some performance criterion. Secondly, the tickets need to be distributed among the probes in such a way that it increasing the probability of finding a feasible low-cost path. In the TBP scheme, the yellow tickets are distributed along low-delay paths thus resulting in a high success probability of finding a feasible path. The strategy for distributing the green tickets is to favour low-cost paths, therefore, obtaining paths with smaller costs which may or may not satisfy the end-to-end requirement. It should be mentioned that the more tickets issued at the source node, the more likely a

feasible path(s) can be found but with a trade-off of introducing more message overhead into the network. On the other hand, issuing tickets economically reduces the chances of finding a feasible path(s). Therefore, this scheme penalizes the events if no feasible paths are found, and since the connection request is rejected then there is neither message overhead nor reward generated from such action.

Chapter 4

Simulation Model and Performance Parameters

4.1 Introduction

In this chapter, we develop a simulation model of QoS routing in order to evaluate the performance of the proposed algorithms. The simulation developed is assumed to emulate the real Internet under different scenarios. However, since routing is a network layer entity and due to the varying performance of routing algorithms with underlying network topologies, we discuss in this chapter different aspects of network graph models and network topologies. Furthermore some adopted parameters and performance measures for simulating the proposed algorithms are also discussed.

4.2 Graph Model

Lately there has been much interest in simulating a more realistic topology to emulate the real internet topology. This is due to the fact that the performance of

routing algorithms may give imprecise results if the evaluation carried out on inappropriate topology. Moreover it is difficult to model a structure for the internet because of its rapid evaluation [81]. Networks can be categorised according to topological properties and its characteristics can be summarized into degree of a node, clustering of a node and shortest-path length between any two nodes [82]. Many existing models can be used in simulating routing algorithms that need some or all the above characteristics. However, we shall briefly describe the models that are relevant and commonly used in this thesis.

4.2.1 Random Topology

Many early efforts to create a more realistic network follow [83], which are based on random graphs. The random graph model is created by adding links with probability of a function of the Euclidean distance between any pair of nodes in the graph. Different graph models produce different distributions of probability on graphs. The most important random graphs will be discussed which are the Waxman [84] and the Doar-Leslie [85] graph models.

4.2.1.1 Waxman graph

The random network topology generator Waxman model proposed for the growth of computer networks. In a Waxman graph the nodes of the network are uniformly distributed in a 2-dimensional Cartesian coordinate space and links are added

according to probabilities that depend on the Euclidean distance between the nodes. The probability to add a link between nodes *m* and *n* is given by [84]:

$$P(m,n) = \beta e^{-d(m,n)/aL}$$

Where $0 < \beta$, a <= 1, d is the distance from m to n, and L is the maximum distance between any two nodes in the graph. An increase in the parameter β will increase the probability of links between any nodes in the graph, while an increase in parameter *a* gives a larger ratio of long links to short links. Although Waxman is widely used and simple to implement, it has one major drawback in that as the number of nodes increases the nodes require impractical node degrees.

4.2.1.2 Doar-Leslie graph

Doar and Leslie proposed a modified model of the Waxman random model so as to limit the average node degree increase. Doar and Leslie added a scaling factor (*KD/N*) to Waxman's link probability equation to stabilize the average node degree [85]:

$$p(m,n) = (KD/N)\beta e^{-d(m,n)/aL}$$

Here k is the scale factor and D is the mean degree of the node.

In Chapters 5, 6 and 7, we used the Doar-Leslie model to generate random network topologies for simulation. Figure 4.1 shows a 40-node random topology generated using the Doar-Leslie model.

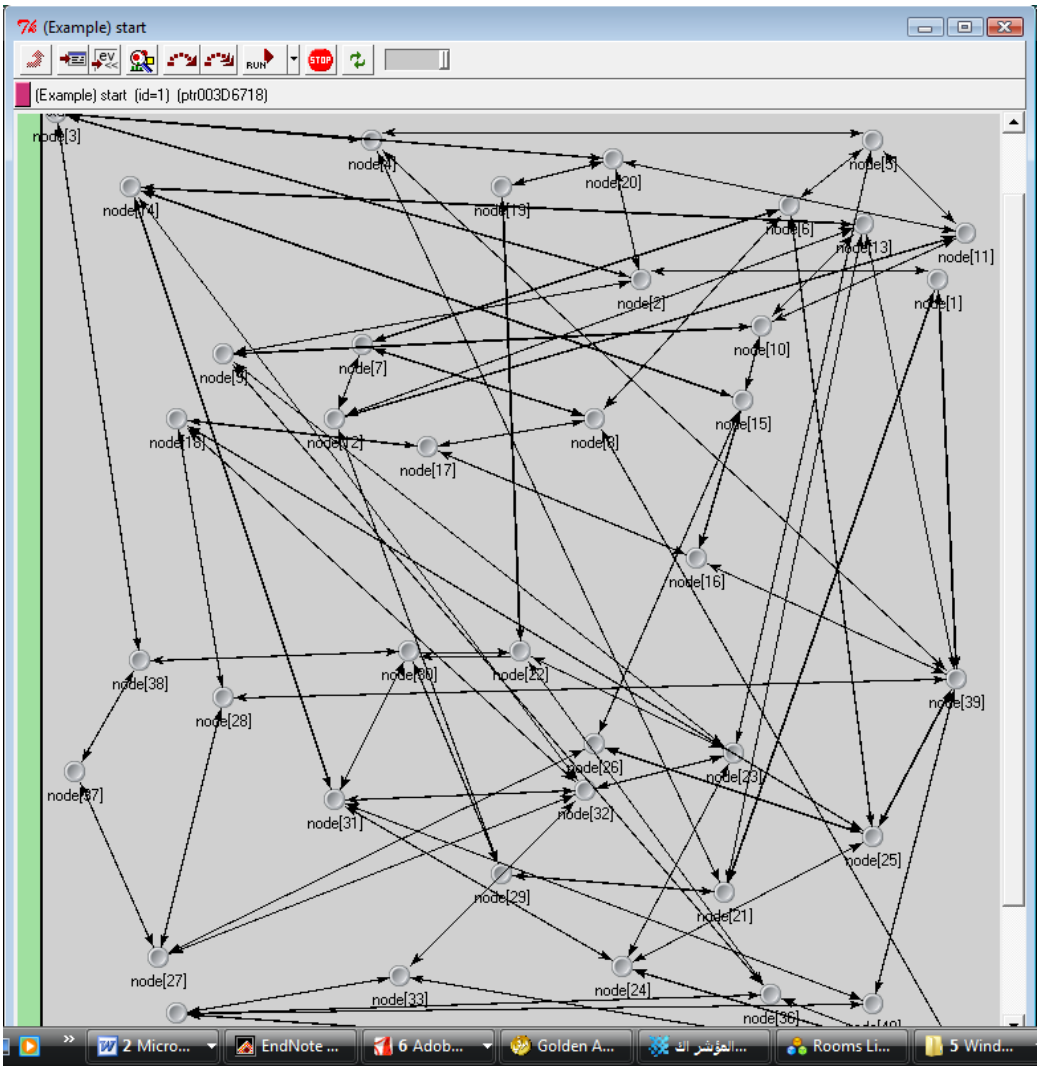


Figure 4.1 Random 40 topology

4.2.2 Regular Topology

A regular graph is a graph where each node has the same number of neighbours. A regular graph with nodes of degree *k* is called a *k*-regular graph or regular graph of node degree *k*. Although a regular graph topology is not common in the internet

it is regularly used to test some features of internet performance. Lattice, torus, star and ring are some example of regular graphs that are used to simulate and test routing algorithms. A lattice is a graph in which the nodes are placed on a grid and the neighbouring nodes are connected by a link. A 4-node torus and lattice graph can be seen in the following figure:

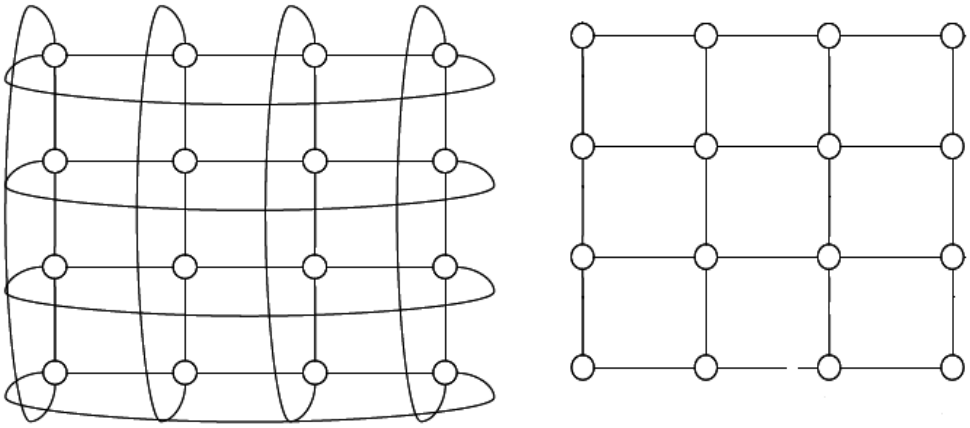


Figure 4.2 a 4-node torus and lattice graph

In Chapters 5 and 6, we use the 7×7 node Lattice and Torus topologies and 3×3 node Lattice in chapter 7 for the simulation. Lattice and Torus topologies have been studied widely to evaluate the performance of QoS routing as they give a variety of path lengths for many source-destination pairs.

4.2.3 Hierarchical Topology

The hierarchical topology is one of the vital computer network models used to evaluate routing algorithms. Hierarchical routing is the key to scale a network

successfully, in which nodes in the same area are connected together and these areas are tied as groups [86] [54]. In Chapters 7, three hierarchical models have been developed each of them are compromised of two-level hierarchical networks to assess a proposed hierarchical algorithm.

4.2.4 ISP Topology

The ISP topology, shown in figure 4.3, represents a single autonomous system domain for Internet Service Provider's networks in USA. The ISP topology has been extensively used in the simulation of routing algorithms [87] [42].

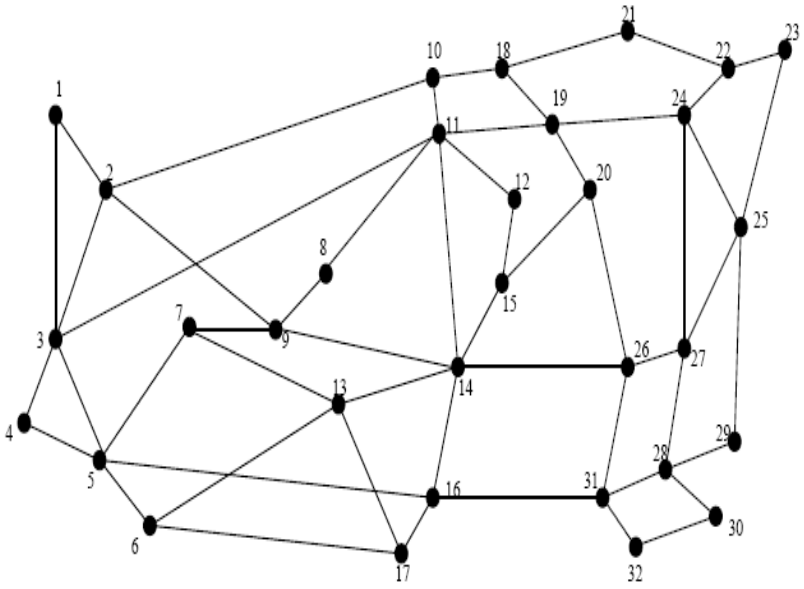


Figure 4.3 the ISP topology

4.3 Simulation Environment

To evaluate the performance of the proposed algorithms the open-source simulation environment OMNeT++ is used [88]. OMNeT++ (Objective Modular Network) is an object-oriented, modular discrete event simulator with an embeddable simulation kernel and GUI support. An OMNeT++ simulation is built on C++ foundations and built out of hierarchically nested modules. Modules are programmed in C++ and use messages as means of communication with each other. A node maintains an arbitrary amount of gates that are used to send messages through links to other nodes. A network topology that contains gates, links and modules, is defined in the Network Description (NED) language.

4.3.1 Simulator Design

Modeling large-scale networks is an issue that must be addressed in designing a network simulator environment. So it was one of our goals to design a simulator that is capable of simulating localized schemes with a relatively large number of nodes. Although the ns-2 simulator package is well-suited for packet switched networks and because of its focus on low level modelling such as packet-level, so it is used for small scale simulations. Moreover, the outstanding number of simulation events that grow linearly with the number of packets can lead to performance bottlenecks when managing a sorted event list of millions of events. For these reasons, the simulator should be designed from the beginning with the

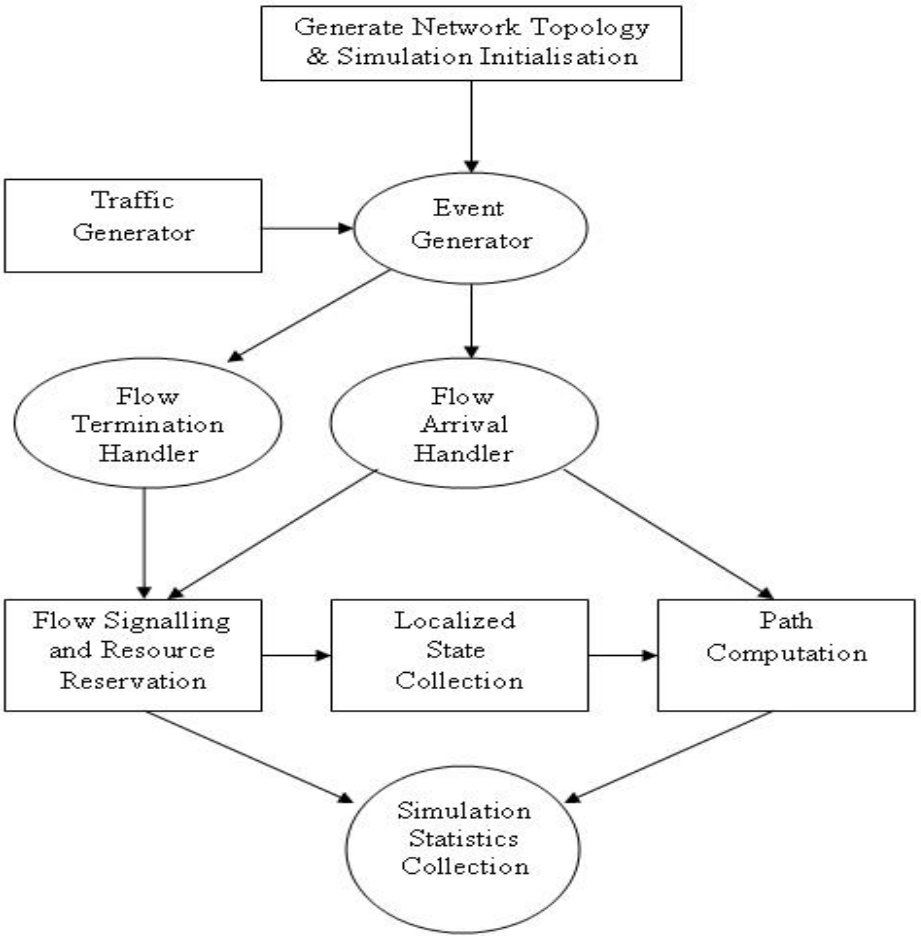


Figure 4.4 Functional components of localized simulator

scalability in mind. We developed our simulator on top of the OMNeT++ that provides a rich functionality for statistical analysis tools for simulating the elements of a communication network, such as nodes, links and packets[89] [90].

4.3.2 Simulator structure

Figure 4.4 and 4.5 illustrate the functional diagram of the localized and the global simulators respectively. The functional divisions are the model components of the

localized and global simulation procedures. Each functional block shown on the above diagrams performs particular functions in the simulation as illustrated below.

Generate network topology and simulation initialization

In this block the variables used in the entire simulation process are initialized. It is started by generating a random topology or reading the regular topology from the NED file. The random topology is generated using the Doar-Leslie model as follows. First the required numbers of nodes are placed randomly across a rectangular coordinate and ensure that nodes are at a least certain distance *(d)* apart. The Euclidean distance *d (u, v)* between nodes u and v and the maximum distance between any two nodes *(L)* in the network are also computed. Then the probability of adding a link (*u*,*v*) is calculated according to the probability $p(m,n) = (KD/N)\beta e^{-d(m,n)/aL}$.

The initial values are assigned to the topology such as link capacities and link delays and other simulation parameter values. Routing tables for all nodes in global algorithms and the set of candidate path for each pair of nodes in the localized algorithms are also constructed in this step.

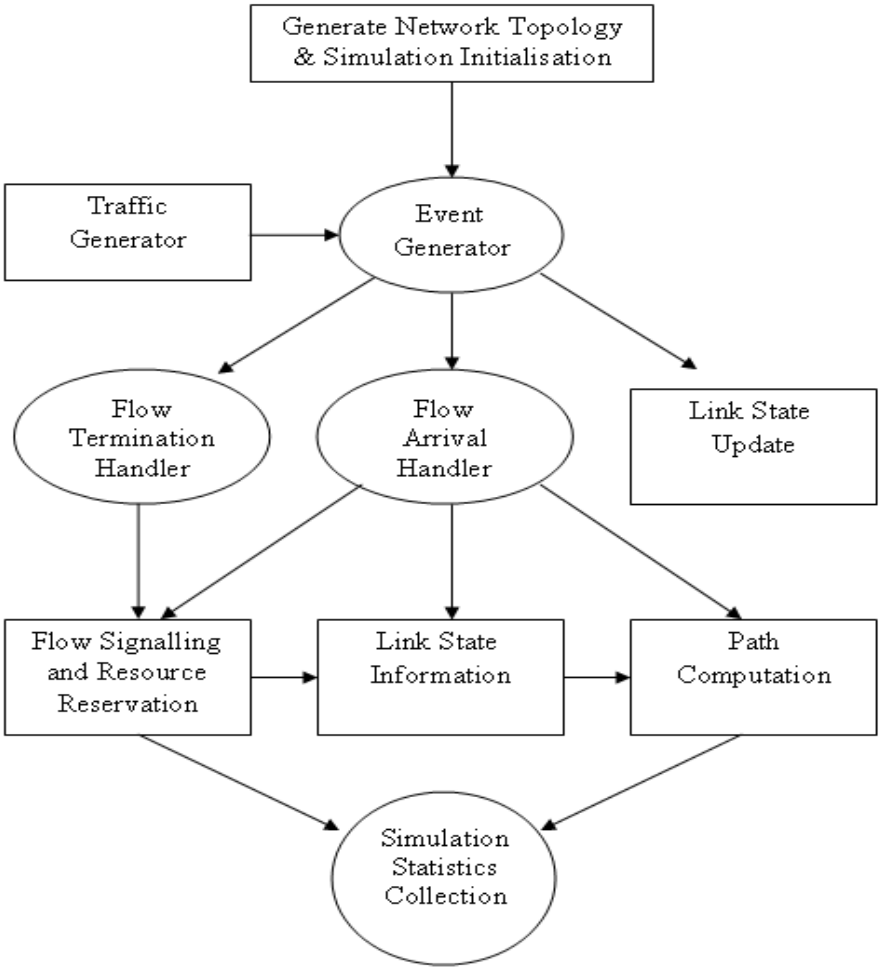


Figure 4.5 Functional components of Global simulator

Traffic generation

The traffic generator model is responsible for specify the characteristics of the traffic in the network. When a connection request arrives to a source node the traffic generator provides the connection request with a random destination node and the flow duration for that flow. The arrival of the connection requests can be modeled as a Poisson stream or bursty stream with different shaper. The

connection request requirements are also specified in this model such as the requested bandwidth and delay constraints.

Event generation:

In response to a flow arrival the event generator can performs three main events:

- Flow arrival handler: this is the main event handler that handles each new connection request and its requirements. It passes each connection request to the path computation module to compute the best feasible path for that connection; once the feasible path is determined the flow signalling module is invoked to signal and reserve resources for the flow.
- Flow termination: this module releases the reserved resources to terminate the flow using the flow and signalling resource reservation module.
- Link state update: this module is used when a global algorithm requires global QoS state information. Link state update information is periodically exchanged among network nodes to obtain a global view of the network QoS state. The QoS state of a link may represent the available bandwidth or the delay since the last update. Based on this current global view of the network state, the path computation module determines a feasible path for a connection request.

Localized state collection

This module is responsible for collecting statistics about network state for localized algorithms. A source node uses locally collected flow statistics generated from itself such as flow arrival rates, flow departure rates and flow

blocking probabilities, and routes traffic based on this local information. This model interacts with the signaling and resource reservation model in order to infer information about the network state.

Path computation model

This model implements various routing policies for localized and global routing algorithms. It uses the link state update information module to find the best feasible path, while in a localized routing algorithm it uses the localized state collection model to route traffic through the most appropriate candidate path.

Flow signalling and resource reservation module

This module is implemented for resource reservation, admission control and signalling policy. This model is triggered when the path computation model finds a feasible path that satisfies the QoS requirements. The signalling process is initiated hop-by-hop from the source node to reserve network resources for the connection arrival.

- Bandwidth

Suppose the requested bandwidth b and each link ℓ in the network has the available bandwidth $bw(\ell)$. As the signalling message passes through the selected path p, each node carries out an admission check over the outgoing link to make sure it has sufficient bandwidth. If the available bandwidth over the outgoing link is equal to or greater than the requested bandwidth the node reserves the bandwidth b for the new flow so that $bw(\ell) = bw(\ell) - b$ and the message is passed

to next node in the path. This module accepts the flow if all links along the selected path *p* have enough bandwidth, if not, a failure message is transmitted back to the source node releasing the reserved bandwidth so that bw(ℓ) = bw(ℓ) + *b* and the flow is rejected.

- Delay:

Suppose the delay constraint is *D* and each link ℓ in the network has delay d (ℓ). As the signalling message passes through the selected path *p*, each node carries out an admission check over the outgoing link adding its delay to the previous delays to make sure that the flow will not be delayed more than the requested delay constraint. If the delay over the outgoing link is less than the requested delay constraint the message is passed to next node in the path. This module accepts the flow if the delay along the selected path *p* is less than the requested delay constraint so that $\sum_{i \in n} d(i) \leq D$, where n is the number of links along the selected path and also delay constraint of any existing flow is not exceeded. Otherwise, a failure message is transmitted back to the source node releasing the reserved resources and the flow is rejected.

This model is also interacts with the flow termination model once the flow duration of a flow has elapsed to release the resources reserved by that flow. It should be noted that this module does not reroute the flow to an alternative path when a failure setup message occurs and as a result the flow is rejected. Although

rerouting flows may decrease the probability of blocking it would also increase the signalling overhead.

Simulation statistics collection

This module monitors large numbers of statistics during the simulation runs. The statistics collection module is invoked by different simulator modules in order to collect different aspects of the performance metrics.

4.3.3 Performance metrics

4.3.3.1 Blocking probability

In QoS routing a path is accepted if and only if it satisfies the required QoS. If the QoS is not satisfied the path cannot be used and so is rejected. In this latter case, valuable network resources have been used in path computation and therefore this is an undesirable overhead. A direct measure of this overhead is the ratio of the number of paths rejected to the total number of connection requests during some long time interval. This gives an estimate of the flow blocking probability which is therefore also a measure of the efficiency of the QoS routing algorithm used.

Flow blocking probability and bandwidth rejection probability are used to measure the performance of the proposed algorithms. Flows will be rejected when one of the links along the path from source to destination does not satisfy the requested bandwidth or the delay over the selected path exceeds the requested delay constraint. The blocking probability is defined as:

Flow blocking probability= $\frac{\text{No of rejected requests}}{\text{No of requests arriving}}$

Bandwidth rejection probability = $\frac{\sum_{i \in B} bandwidth(i)}{\sum_{i \in C} bandwidth(i)}$

Here B is the set of blocked flows and C is the set of total flows. Bandwidth (i) is the requested bandwidth for path i.

4.3.3.2 Load balancing

Load balancing is considered an important factor to measure the performance of QoS routing algorithms. The primary objective of load balancing is to use the network resources in a more efficient manner in order to reduce the risk of network traffic congestion. A Load-balanced network should result in less delay and packet loss than one with an imbalanced load.

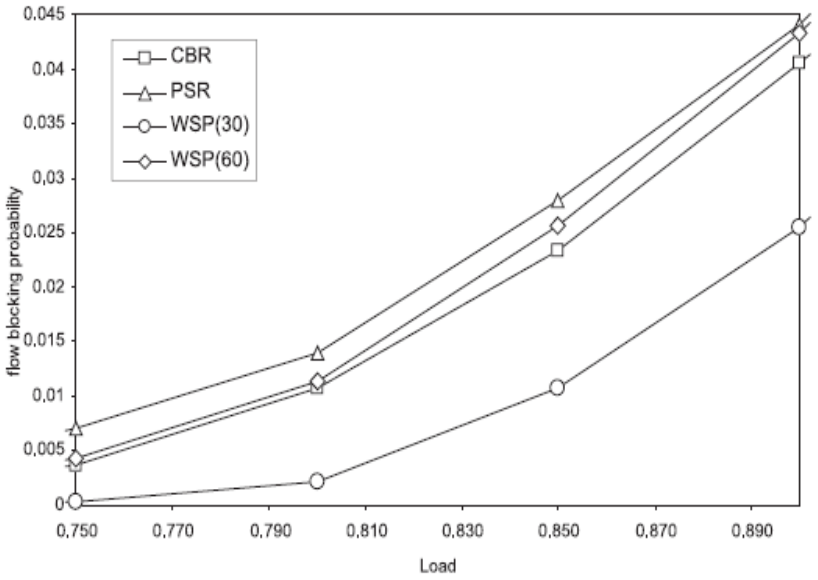
In general, it is important in designing QoS routing algorithms to fairly distribute the load among links in network topologies. An important metric of evaluation of a QoS routing algorithm is therefore the load balance where the fairness of load is computed in the network to measure the efficiency of routing algorithms. In our fairness calculation, the well-known Jain's fairness index [91] is used to evaluate the fairness of the algorithms. Let N be the number of links in the network; we define Jain's index for a set of links $\{x_1, x_2, .., x_N\}$ as:

$$Jain's\ Index = f(x_1, x_2, ..., x_N) = \frac{\left(\sum_{i=1}^{N} x_i\right)^2}{N\sum_{i=1}^{N} x_i^{\,2}}$$

Jain's index is bounded between 1/*N* and 1; when the value is 1 the algorithm is 100% fair, whereas a value of 1/*N* is the least fair.

4.3.4 Simulator validation

Validation guarantees that the simulation behaves as expected by ensuring that there are no significant differences between the simulator model results and ones known to be correct [92]. Simulator model validation is the method used to prove that the results obtained are correct. For this thesis, some validation testing has been conducted through the simulations in OMNET++ to verify the correctness of the localized and global QoS routing algorithms. To validate the credit based routing algorithm (CBR), the results obtained have been compared to those obtained by [80]. Similarly for the WSP algorithm, using the same simulation parameters and configurations described in [80] by the designers of the CBR algorithm, we repeated the simulations using our simulator and found the result are very close to the results reported shown in Figure 4.6.



(a) Original results. (Taken from [80]).

(b) Verified results

Figure 4.6 Simulator Validation Results

4.4 Summary

In this chapter we have developed a graph model and the simulation model that is built on the top of OMNET++ to evaluate the performance of the proposed algorithms in chapters 5, 6 and 7. This chapter also described the simulator design and how it was validated. Different types of network topologies, parameter settings and the performance metrics such as blocking probability and load balancing used in the performance evaluation were also described.

Chapter 5

Localized Bandwidth Based QoS Routing

5.1 Introduction

Routing in the internet was not originally planned for optimizing the performance of the network since it relies mainly on shortest path mechanisms. Although many QoS routing algorithms have been proposed to optimize the performance of network usage, they suffer from some drawbacks. Most of the proposed QoS routing algorithms require maintaining knowledge of network global state, and using this knowledge to compute the QoS path. However, the primary drawback of QoS routing is the scalability in large networks, and this is because of the need for each node to collect global state information about the complete network state. The other type of QoS routing schemes, which are localized QoS routing, do not maintain global network state information, but instead infer the network QoS state from locally collected flow statistics. These types of scheme avoid the overhead

messages that are exchanged in order to keep up to date global state information and therefore avoid the routing scalability problem.

The success of the localised routing in telephone networks and the limited number of promising researches in the area of localised QoS routing has motivated the present author to search for other viable localised methods.

Since the DAR algorithm is planned for telephone networks and CBR outperforms PSR we focused on the CBR scheme. Although, CBR shows good performance against the other localized QoS routing PSR, the criteria used for path selection in the algorithm, which relies on a crediting scheme, does not directly reflect the quality of a path, which is only reflected indirectly by the addition or subtraction of credits. It is conjectured the quality of a path should be measured directly based on the required QoS metric like bandwidth in CBR. Moreover, there does not appear to be any analytical justification in using blocking probability as an increment or decrement factor for the credits, but the justification appears entirely intuitive.

In this chapter we propose a bandwidth based routing (BBR) which is a simple localized QoS routing algorithm that relies on average residual bandwidth on the path in order to take routing decisions. We study other localized QoS routing schemes: firstly, the localized Credit Based Routing (CBR), proposed in [80]; and the global QoS routing scheme Widest Shortest Path (WSP), proposed in [45]. We

compare their performance with our scheme in terms of flow and bandwidth rejection probability under different network loads and topologies.

5.2 Bandwidth Based Routing

The localized bandwidth-based routing (BBR) algorithm relies on the average residual bandwidth in the path in order to take routing decisions. Unlike PSR and CBR, which performs routing decisions based on flow statistics of path blocking probability, BBR uses a completely different scheme in terms of path selection based on the collection of statistics about the residual bandwidth in each candidate path. The average residual bandwidth for each candidate path is calculated and then used to measure the quality of the path, and upon flow arrival the path with the highest average residual bandwidth is used to route the incoming flow. BBR keeps monitoring the residual bandwidth in the network and continuously updates each path's average residual bandwidth in the candidate path set. Using the average residual bandwidth gives the actual quality of the path.

The BBR is a source routing algorithm where the source node takes the routing decision. When a new connection arrives, the source node computes the path that may satisfy the QoS bandwidth requirement. It uses a setup message to travel along the selected path with each connection request. Each intermediate node performs an admission test for the outgoing link residual bandwidth to check the

ability of the link to satisfy the requested bandwidth. If the link has sufficient bandwidth, the requested bandwidth is reserved for that connection and the message is sent to the next hop. The network admits the connection if all links along the path can satisfy the requested bandwidth for the flow duration. However, a failure message occurs if any link along the path does not support the requested bandwidth. Messages are sent to the source node to calculate the average residual bandwidth for that path. The pseudo code for the BBR algorithm is as follows:

PROCEDURE BBR ( )

Initialize

Set Pavg =CAPACITY, $\forall$ P $\in$ R

BBR ( )

1. If Pavg =0 $\forall$ P $\in$ R
2. Set Pavg =CAPACITY, $\forall$ P $\in$ R
3. Set P=max {Pavg: P $\in$ R}
4. Route flow along path P
5. If flow accepted
6. Calculate Average Residual Bandwidth (P)
7. P.Residual Bandwidth =min {Link. Residual Bandwidth: Link $\in$ P}
8. Value= (value + P.Residual Bandwidth)

9. Pavg= {Pavg, Value}
10. Else
11. Calculate Average Residual Bandwidth (P)
12. P.Residual Bandwidth =min {Link. Residual Bandwidth: Link ∈ P}
13. Value= (value - P.Residual Bandwidth)
14. Pavg= {Pavg, Value}

END PROCEDURE

Each source-destination pair in localized bandwidth-based QoS routing requires a predefined set of candidate paths R. The main characteristic that is associated with every path P in the candidate path set is the average residual bandwidth. We use Pavg to store the average bandwidth and update its value with every connection request. Upon flow arrival, BBR selects the path with the largest average residual bandwidth (line 3) and routes the flow along the selected path. If the flow is accepted along the accepted path, the residual bandwidth is calculated along it. As the setup message travels to the destination it performs a comparison over the links along the path to get the minimum residual bandwidth for that path (line 7). The path residual bandwidth is then added to previous values of the path and stored in the source node. As a new connection arrives to the source node, the stored values are divided by the number of connections sent in order to estimate the actual average bandwidth of the path. It should be noted that choosing the

minimum residual bandwidth among the links on the selected path reflects the actual bandwidth that the path can support. In contrast, if the flow is rejected its residual bandwidth is subtracted from the overall path residual bandwidth and the new average is calculated as previously. When the path's residual bandwidth is decreased its probability to be chosen is also decreased for new connections. Increasing or decreasing the path bandwidth by residual bandwidth reflects on the actual path state and the quality of the path can be measured accurately.

Unlike PSR and CBR, which monitors flow blocking probabilities, BBR monitors bandwidth of a path and the source node stores positive and negative residual bandwidth values for each accepted or rejected flow of each path respectively. It calculates the average bandwidth using a simple moving average (sliding window) over a predetermined period. So, for the sliding window W, the average bandwidth will be calculated over the most recent W connection requests. For example, if B= {1,-1.2, 3} represents the last three residual bandwidths collected over a period W=3 for path P, the average bandwidth that the path P could support would be (1-1.2+3)/3=0.933. Suppose a new arrival is rejected and the residual bandwidth of the path was 0.8; the set B will be changed to B= {-1.2, 3,-0.8} and the new average bandwidth would be (-1+3-0.8)/3=0.4.

5.3 Performance evaluation

This section evaluates the performance of the proposed localized bandwidth-based routing scheme (BBR) and compares it with the localized CBR scheme. We do not use PSR for comparison since CBR has been shown to outperform PSR in almost all situations [2]. The global routing algorithm, Widest Shortest Path (WSP), was also used in the comparison. WSP finds the minimum hop count path that satisfies the bandwidth constraint. If there is more than one path with the same length, the one with the maximum available bandwidth is selected. [We use the notation WSP (x) to refer to this algorithm with update interval of x time units for link state information]

5.3.1 Simulation model

We implemented our localized bandwidth-based QoS routing scheme (BBR) based on the discrete-event simulator OMNeT++ [90] [88] and conducted extensive simulations to test its performance. Using one of the predetermined algorithms (BBR, CBR, and WSP), the simulation performs path selection, resource reservation, and admission control at flow level.

Due to the varying performance of algorithms with underlying network topologies[93], we have used different types of network topologies. We used an ISP topology in the simulation, which is widely used in different QoS routing algorithm studies [93] [94] . A 49-node torus regular topology was also used in

the simulation to be able to select different path lengths between each source-destination pair. Random topologies were created using C++ and OMNeT++, where the connection between any two nodes is determined by a probability using the Doar-Leslie Model [85]. This model is based on the Waxman Model using a scaling factor, which stabilizes the node degree of the graph. Table 5.1 lists the most important characteristics of the topologies used in the experiments.

Topology	**Nodes**	**Links**	**Node degree**	**Avg. path length**
RANDOM80	80	484	6.07	2.92
RANDOM40	40	156	3.90	2.77
ISP	32	108	3.37	3.17
Torus	49	196	4	3.50

Table 5.1: Network topologies and their characteristics

5.3.2 Traffic generation

In each experiment in the simulation the network topology remains fixed. The traffic is generated by having the source node choose the destination node from amongst all nodes except the source node with uniform probability. Flow interarrival times at a source node are exponentially distributed with the mean $1/\lambda$. The mean flow holding time is $1/\mu$ which is again exponentially distributed,

while the QoS requested bandwidth is uniformly distributed in the range between 0.1 to 2MB. We also assume that all links in the topologies are bidirectional, each with the same capacity C in each direction (C=150Mbps).

Following [1] , the offered network load is $L_d = \lambda Nhb/\mu LC$, where N is the number of nodes, b is the average bandwidth required by a flow, h is the average path length in number of hops (averaged across all source-destination pairs.) and L is the number of links in the network. The parameters used in the simulation for CBR are MAX_CREDITS=5 and Φ=1. Blocking probabilities are calculated based on the most recent 20 flows.

The candidate path set is computed based on the current network topology and then recalculated in part whenever a topology change occurs. In fixed wired networks this would be expected to happen only infrequently. This might be done for each sub-network and backbone separately with the topology information stored in an array of links and nodes. For the purpose of the simulation the set of candidate paths between candidate paths between each source-destination pair in a selected network topology are chosen, so we include minimum hop and (minimum hop) +1 in the set to get the required number of candidate paths between each pair. This process starts by assigning an initial value 1 to all links in the network and then uses the Dijkstra algorithm to find the shortest candidate path. After finding the first candidate path, the weights on all the links along the

path are increased and repeats the step to find the next candidate path until no more new paths can be found.

All experimental results collected are based on at least 3,000,000 connection requests (arrivals) and the results are collected after 200,000 connection requests to allow a steady state to be reached. Each experiment was repeated 20 times with confidence interval at 95% confidence level and found that most of the confidence intervals were not visible on the figures(see figure 5.1 (a)).

5.3.3 Performance metrics

Flow blocking probability and bandwidth rejection probability were used to measure the performance of the algorithms. Flows are rejected when one of the links along the path from source to destination does not satisfy the requested bandwidth. The blocking probability is defined as:

$$\text{Flow blocking probability} = \frac{\text{No of rejected requests}}{\text{No of requests arriving}}$$

$$\text{Bandwidth rejection probability} = \frac{\sum_{i \in B} bandwidth(i)}{\sum_{i \in C} bandwidth(i)}$$

Here, B is the set of blocked paths and C is the set of total requested paths, and bandwidth (i) is the requested bandwidth for path i.

5.3.4 Simulation results

5.3.5 Blocking probabilities

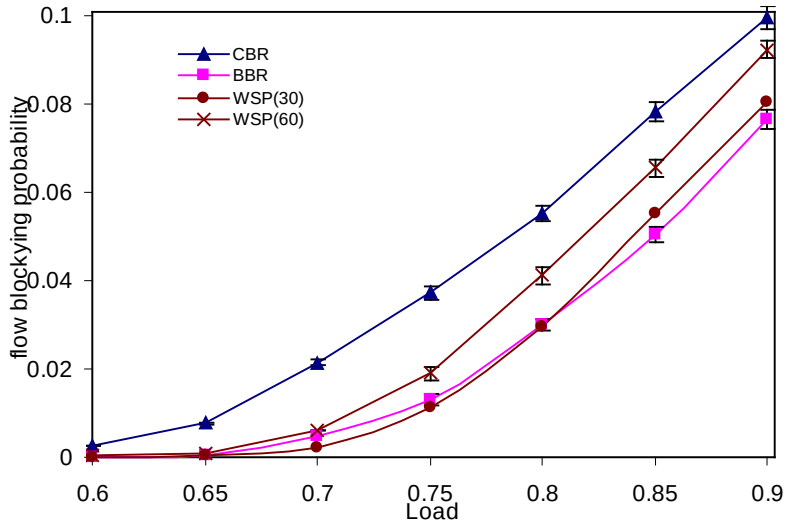
The performance of CBR, BBR, and WSP is compared under different settings using flow blocking probability and bandwidth rejection probability.

5.3.6 Impact of various load conditions

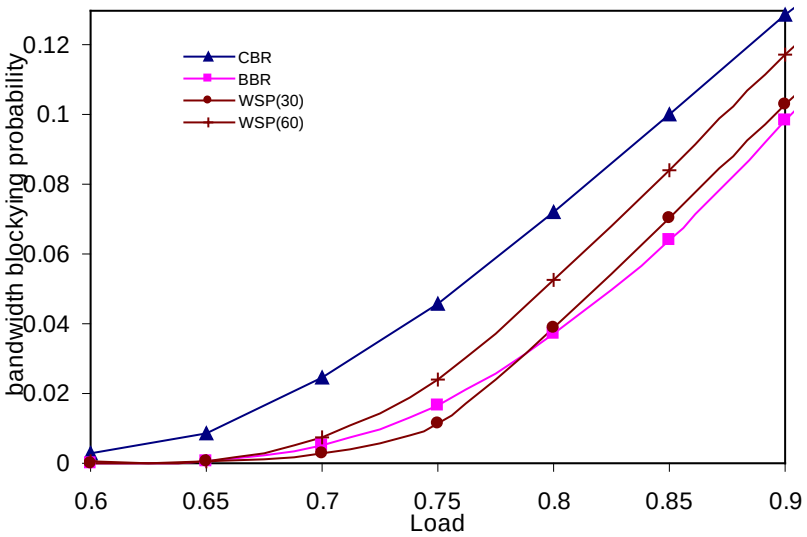
Figure 5.1 compares the performances of CBR, BBR, and WSP algorithms by measuring the blocking probability under various loads. The load is uniformly offered between all the 40 nodes in a random topology. Flow and bandwidth rejection probability of CBR, BBR, and two update intervals of WSP, 30 and 60, are plotted against varied offered load from 0.6 to 0.9. It can be noticed that most of the flows below 0.6 loads are accepted for all the three algorithms. As the load increases over 0.6 the blocking probability gradually increases for WSP (30), WSP (60), and BBR; whereas the CBR is significantly increased. In fig. 1(b), when the load increases the bandwidth rejection dramatically increases as it is difficult to satisfy the large bandwidth QoS. We can see that CBR gives the worst performance in terms of flow and bandwidth rejection probability, even under low load. Update intervals of WSP significantly affect its performance as WSP (30) gives lower blocking probability than WSP (60) because of its updated view of the global network state and its periodic update responding quickly to changes in

the link state. Since BBR takes routing decisions based on the residual bandwidth which is monitored frequently with each connection request, its blocking probability is the least among the other algorithms. It even performs better than WSP (30), particularly under heavy load.

WSP always tends to select the shortest paths, even when they are congested, until its current view of the network state is updated, whereas CBR selects the path with the largest credits which is changeable according to blocking probability. This leads the CBR to select alternative paths. In the same way BBR selects paths with the largest average residual bandwidth, which gives more scope to select paths, as long as they satisfy QoS bandwidth. It is also notable that there is virtually no difference between the results for bandwidth or flow blocking probability, suggesting that either may be used as a performance measure.



(a) Flow blocking probability



(b) Bandwidth rejection probability

Figure 5.1 Flow and bandwidth blocking probabilities in Random 40.

5.3.7 Impact of bursty traffic

Although Poisson traffic is widely used to model network flow arrivals, we also model bursty traffic in the Random 40 topology, as some studies [95] [96] showed

that the flow arrival is bursty and that distributions with heavy tails are more realistic traffic. Following [1] [96], the burstiness of traffic is modelled using a Weibull distribution with two different values of the shape parameter of the distribution, 0.3 and 0.7, where burstiness is increased with a smaller shape value. Figure 5.2 shows the bandwidth rejection probability plotted against the offered load from 0.6 to 0.85 with different shape values. It can be noticed that the more burstiness in the network arrival, the more blocking probability. The performance of CBR is poor, particularly in the case of shape 0.3, where the burstiness is increased. This is obvious, since CBR takes routing decisions based on blocking probability, which is increased with burstiness. However, BBR and WSP (60) with shape 0.7 give superior performance.

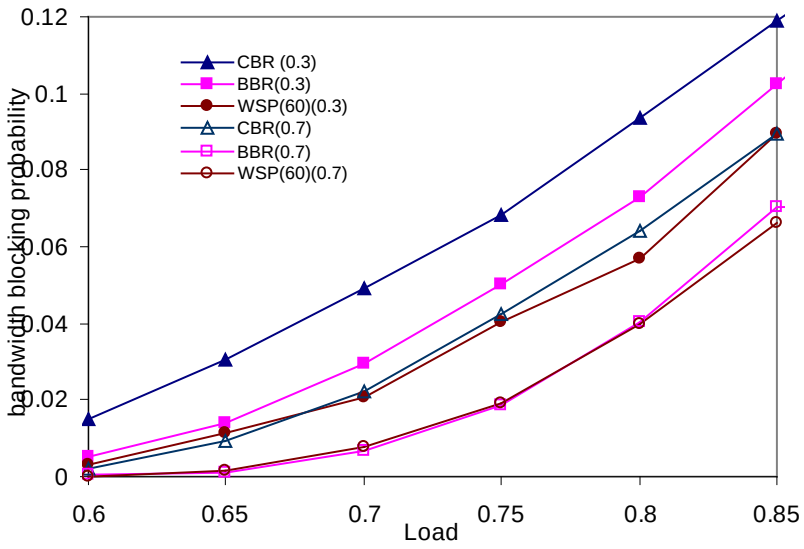


Figure 5.2 Impact of bursty traffic in Random 40.

5.3.8 Impact of large bandwidth

We consider two types of bandwidth traffic in order to study the impact of large bandwidth flows and small bandwidth flows but having the same holding time. The amount of bandwidth requested for both types are uniformly chosen from the range (2-4) for the large flows and (0.1-2) for the small flows, with the mean *b2*=3 and *b1*=1.05 respectively. The holding times for all flows are exponentially distributed with the mean 160.4. The performance is measured by mixing fractions of small and large flows *f* and *(1-f)* respectively, keeping the offered load fixed at L_d =0.8. The arrival rates are changed using the following formula:

$$L_d = \frac{\lambda(f \times b_1 + (1-f) \times b_2)Nh}{\mu LC}$$

Figure 5.3 shows the bandwidth rejection probability plotted against the fraction of small bandwidth flows. It can be noticed that CBR gives poor performance, regardless of the fraction of small bandwidth; and unlike WSP (30) that gives the best performance. BBR performs better than WSP (60) with small flows, which can be noticed when most of the bandwidth fractions are small. However, in the case of mixing small and large bandwidth with any ratio, WSP (60) performs better. This is expected as BBR continuously monitors the bandwidth and the amount of bandwidth requested is known before path selection for both large and

small bandwidth. However, as the bandwidth varies from large to small bandwidth, the algorithm may not be able to classify it into bandwidth classes and the path that is good for one class of bandwidth (e.g. small bandwidth) may not be good for another class of bandwidth (e.g. large bandwidth). It should also be noticed that the difference in performance remains fixed between CBR and BBR, and they have good performance with small requested flows or large requested flows but not with a mixture.

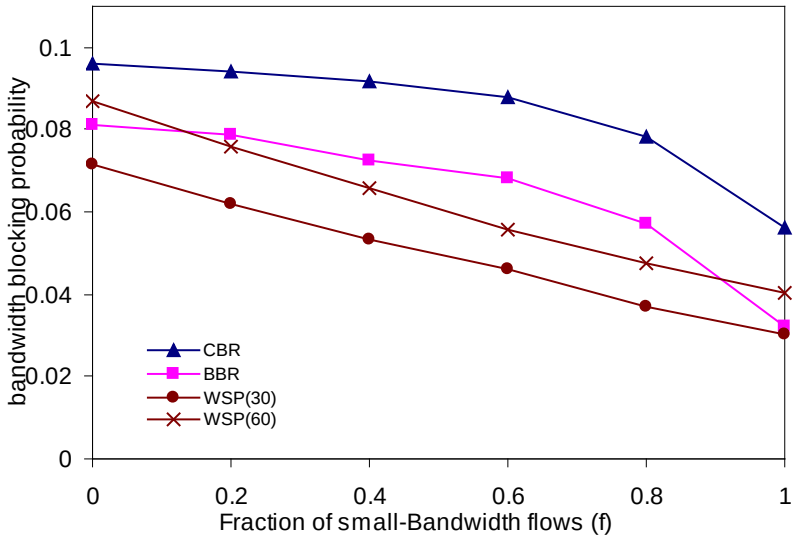
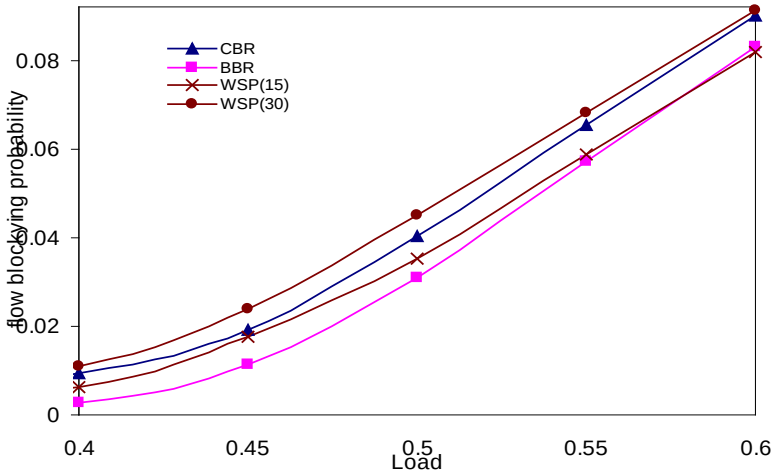


Figure 5.3 Impact of large bandwidth in Random 80.

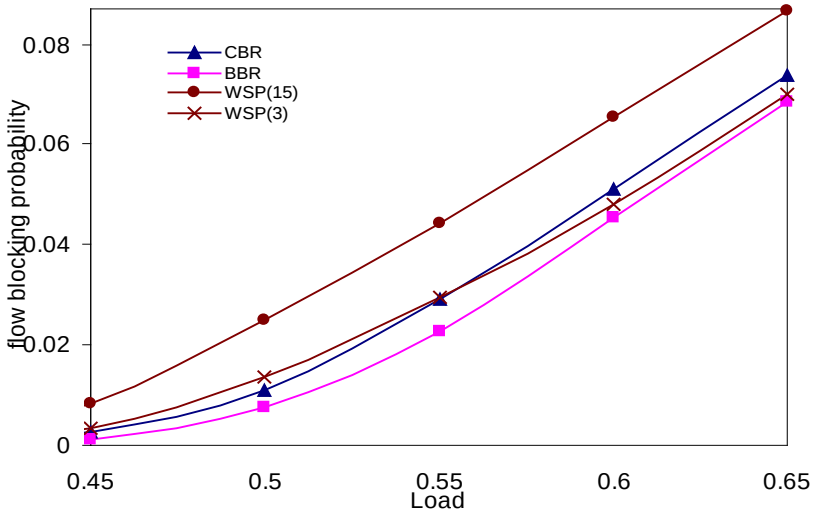
5.3.9 Impact of network topologies

In figure 5.4 the flow blocking probability is plotted against different ranges of offered load for different types of network topologies. The characteristics of these

topologies were described earlier in table 5.1. Generally speaking, it can be noticed that BBR performs better than CBR for all the different topologies. Moreover, BBR also performs well against the WSP algorithm. This can be seen in the ISP topology where BBR performs better than WSP (3), which has a very small update interval. In the case of random 80, our scheme also performs better than WSP (15) for the load $L_d \leq 0.55$, which also has a very small update interval. BBR still gives very good results on Random 40 as in figure 5.1(a) and its blocking probability is better than WSP (60) and WSP (30) for the load $p \leq 0.8$. However, in the case of the Torus Topology, BBR fails to perform better than WSP (30), but still gives good results against WSP (60) and CBR. This is most likely because the Torus is a regular topology and so there would be less likelihood of route flapping with WSP.



(a) Random 80 topology



(b) ISP topology

(c) Torus topology

Figure 5.4 Impact of network topology.

5.3.10 BBR sensitivity to W parameter

Since BBR monitors the residual bandwidth in each path of the candidate path sets, a sliding window W, with a predetermined period, is used to record residual bandwidth upon flow acceptance or rejection. In figure 5.5 flow blocking probability is plotted against the offered load using different values of W connection requests. It was found that the blocking probability decreases as the value of W increases. This parameter controls the observation period of the path bandwidth, so a longer period is needed to get a good estimation on how good or bad the path is. Figure 5.6 shows the blocking probability plotted against window size for Random 40, with the fixed load L_d =0.8. BBR showed that it gives better performance as the value of the window size increases.

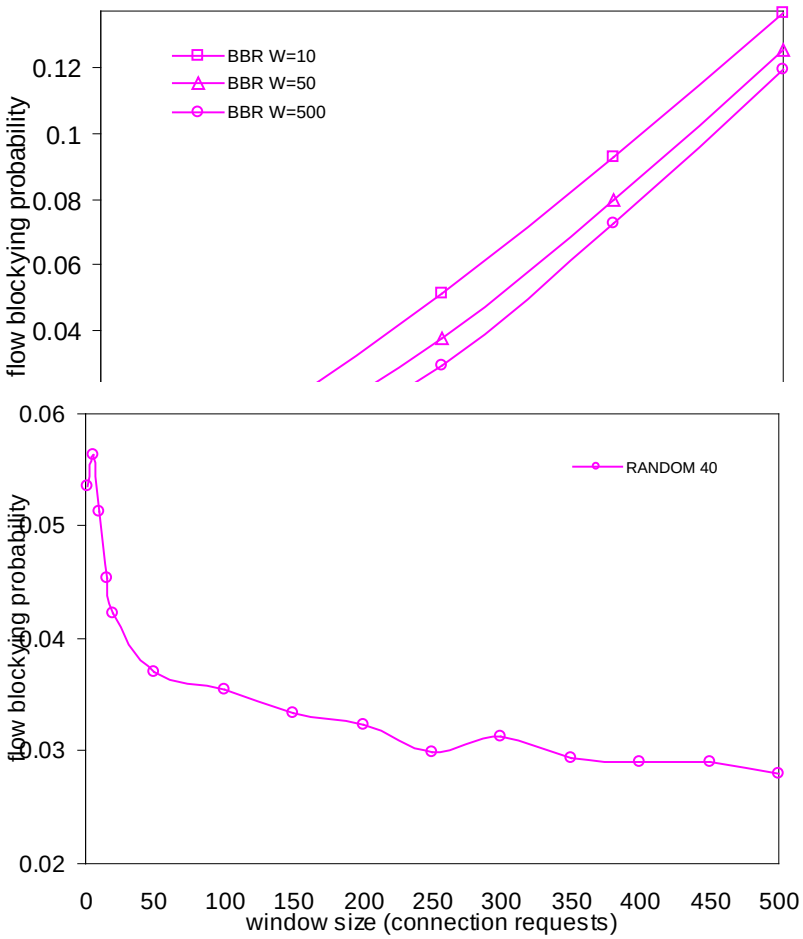


Figure 5.6 Choices of window size in Random 40

That is, the more data collected about the residual bandwidth which results in more accurate information about the quality of the path obtained. So, a reasonably large size of window size reflects the bandwidth that the path can support; small values may not reflect the path quality, since load in the network may change rapidly. This can be seen in figure 5.6, where the blocking probability sharply decreases as the window size increases until it reaches 50. Choosing values for the window size, between 50 and 250, gives a gradual decrease of blocking probability; whereas values larger than 250 have a negligible effect on blocking probability. It should be noted that the window size in this example is appropriate for the random 40 network topology and may not be the same for other topologies. The results obtained here may not necessarily be applicable for other network sizes and topologies. This is also the case for the window sizes used in chapter 6 and chapter 7.

5.3.11 BBR overhead and time complexity

Global QoS routing algorithms performing a variant of Dijkstra's algorithm to find the shortest path or widest path, or both like WSP, take at least O (N log N+L) time, where N is the number of nodes and L is the number of links in the network. On the other hand, the complexity of selecting a path among the set of candidate paths R in CBR and BBR is O (|R|). CBR needs to update blocking probabilities,

which takes a constant time O (1). Similarly, BBR needs to update the average bandwidth, which also takes O (1).

We can claim that QoS routing is scalable when the network overheads increase linearly or better as network size increases. There are several overheads associated with a QoS routing scheme. We categorize the overheads incur by a QoS routing scheme into update overhead and path computation overhead. The global QoS routing schemes performs a variant of Dijkstra's algorithm to find the shortest path with maximum bottleneck bandwidth and thus it search the whole network in order to find a feasible path. On the other hand, in localized QoS routing schemes one of the candidate paths is chosen and time complexity is O(R), which is much less than O (N log N+L) for global schemes. Moreover, the selection of the feasible path is increased linearly with the number of candidate paths R and not exponentially as in many global schemes. When considering the collecting of state information, the global QoS routing at each node requires a view on the status of network throughout link state updates. Every node is required to maintain QoS state and create updates about all the links attached to it. The frequent updates cause both communication and processing overhead on the network and consume both network bandwidth and processing power. On the contrary, the nodes employing localized schemes do not use any update exchange and thus entirely avoid routing overhead. Only source nodes need to monitor statistics about the candidate path set and send a flow to the most feasible path.

Considering that the localized QoS routing schemes perform better than global QoS routing schemes without introducing more complexity at core routers and more communication overhead on network, we conclude that localized QoS routing schemes are more generally scalable than global QoS routing schemes.

5.4 Summary

In this chapter we proposed a bandwidth-based localized QoS routing scheme (BBR) that takes routing decisions using bandwidth statistics collected locally. We have compared it with CBR and the commonly used WSP routing schemes under different traffic loads and network topologies. We have demonstrated through the simulations that our scheme performs better than CBR in all conditions and outperforms the WSP scheme in some network topologies, even for a small update interval of link state. In general our results suggest that:

- Localized QoS routing should be based on schemes that explicitly reflect the quality of a path, rather than schemes that are based indirectly on the path quality. CBR and PSR are based indirectly on path quality through the blocking probability, but are not directly related to residual bandwidth, which is the required QoS metric.
- Also, the path quality should be measured based on averages taken over a moving window of connection requests, and this window should be of a specific minimum size to obtain the best results

Chapter 6

Localized Delay Based QoS Routing

6.1 Introduction

The growth in demand for real time multimedia applications has motivated researchers to develop routing algorithms to accommodate delay as the QoS in the internet. The main goal of the contemporary QoS routing algorithms is to find feasible paths constrained by the application requirements. In source QoS routing algorithms the computation is entirely computed at the source node using global state information. Routing overhead associated with maintaining and distributing the global state information can be high especially when a network is large. However the blocking probability of connection arrivals for delay constrained QoS routing can be reduced by increasing the accuracy of global state information. The balance between the frequency of update intervals and link state information is very important[1]; the more update intervals of link state information the more accurate state information and vice versa. Such high levels of exchange may incur large communication and processing overheads.

In the previous chapter of this thesis, we proposed an efficient localized QoS routing algorithm to overcome the global QoS routing algorithm drawbacks. Despite the fact that the proposed localized algorithm showed simplicity in selection of feasible paths resulting in very low time complexity $O(R)$ with also small protocol complexity (this latter is ignored because it is expected to be small), it does not guarantee the end-to-end delay, which is required for real-time applications.

In this chapter we propose delay-based routing (DBR) which is a simple localized QoS routing algorithm that relies on average delay on the path in order to take routing decisions. We study other QoS routing schemes, such as localized credit-based routing (CBR) proposed in [80]. We develop it using delay instead of bandwidth. We also use the global QoS routing scheme shortest path (Dijkstra) and Shortest-Widest-Path (SWP) [25] algorithm to compare their performance with our schemes in terms of flow blocking probability under different network loads and topologies. No other localized QoS routing algorithms have previously been proposed that use delay as the QoS metric.

6.2 The proposed algorithms

In this section, we propose two new localized routing algorithms. These are considered source routing algorithms, as the source nodes have the ability to select an explicit path to the destination node. The proposed algorithms assume

that the network performs resource reservation and sends signalling messages to set paths for new connections. In our proposed algorithms, end-to-end delay is used as the quality of service metric. The algorithms guarantee that the end-to-end delay which the flow experiences between the source node and the destination node never exceeds a certain constraint value.

6.2.1 Credit Based Routing

The connection signalling in CBR starts when a new connection request arrives at the source node, which sends a setup message along the selected path. The message stores the delay over the outgoing link, and each intermediate node performs an admission test for the outgoing link and adds the outgoing link delay to the previous delay. If the delay that the message experiences is less than the QoS delay, the delay is reserved for that flow and the message is forwarded to the next node. The flow is accepted if the delay experienced in the selected path is less than the QoS delay, as in (3), which means that end-to-end delay over that path satisfies the delay constraint. Otherwise, if the delay over the selected path exceeds the QoS delay, a failure message is propagated back to the source node and the flow is rejected. This means either that the delay over that path does not satisfy the delay constraint and/or the delay constraint on some existing path is exceeded.

$$\sum_{i \in n} delay(i) \leq QoS\_Delay \quad (3)$$

where n is the number of links along the selected path.

The information regarding flow acceptance or rejection is acknowledged to the source node in order to collect flow statistics. The pseudo code for CBR algorithm is as follows:

PROCEDURE CBR ( )

Initialize

Set P.credits = MAX_CREDITS, $\forall$ P $\in$ R

CBR ( )

1. If P.credits =0 $\forall$ P $\in$ R
2. Set P.credits = MAX_CREDITS, $\forall$ P $\in$ R
3. P^{min} =max {P.credits: $P \in R^{min}$ }
4. P^{alt} =max {P.credits: $P \in R^{alt}$ }
5. If P^{min}.credits >= $\Phi \times P^{alt}$.credits
6. Set $P = P^{min}$
7. Else
8. Set $P = P^{alt}$
9. Route flow along path P
10. If sum {L.delay: L $\in P$} $\leq$ QoS_Delay & for all existing paths P` {L`.delay : L $\in P`$ , $P` \cap P = \phi$ } $\leq$ (QoS_Delay)` (flow accepted)

11. UpdateBlockingProbability (P)
12. Amount = (1- *P*.BlockingProbability (P)
13. *P*.credits=min {*P.credits* + amount, MAX_CREDITS}
14. Else
15. UpdateBlockingProbability (P)
16. Amount = (*P*.BlockingProbability (P)
17. *P*.credits=min {*P.credits* - amount, 0}

END PROCEDURE

Note that in step 10 the second term in the AND clause is to ensure that any existing path *P`* that has links in common with the requested path *P* does not have its QoS jeopardised by the new connection.

Like most localized QoS routing algorithms, CBR requires every node to maintain a predetermined set of candidate paths *R* to each possible destination. CBR distinguishes between two types of paths: the set of minhop paths $R^{\min}$ and the set of alternative paths R^{alt}, where $R = R^{\min} \cup R^{alt}$. *P*.credits is a variable associated with each candidate path. $P \in R$ stores credits for each candidate path. *P*.credits is set to MAX_CREDITS, which is a system parameter that determines the maximum credit each candidate can gain. CBR selects two paths upon flow arrival from each set of candidate paths, $P^{\min}$ (line 3) and P^{alt} (line 4), which are

the paths with maximum credits in the minimum candidate paths set R^{min} and the path with maximum credits in the alternative candidate paths set R^{alt}.

The flow is routed along the path with the largest credit P^{min} among the minimum candidate paths set, if P^{min}.credits >= Φ× P^{alt}.credits (lines 5-6); where $\Phi \leq 1$, otherwise, P^{alt} is chosen (line 8). Φ is a system parameter that controls the usage of alternative paths and limits the 'knock-on' effect. The flow is accepted if the end-to-end delay along the selected path satisfies the QoS delay (delay constraint) and the delay constraint of any existing path is not violated (line 10), as described earlier. The blocking probability for the selected path *P*.credits is updated by increasing its credit with an amount that corresponds to its success probability (line 11-13) when the flow is accepted. Whereas if the flow is rejected, the blocking probability for the selected path *P*.credits is updated by decreasing its credit with an amount that corresponds to its failure probability (line 15-17).

CBR uses a sliding window to calculate the blocking probability with a predetermined size, W *(*connection requests). It keeps monitoring the flow blocking probabilities and records them upon flow acceptance and flow rejection. So, for each path of the candidate paths set, the blocking probability is computed for the recent period. The sliding window W moves to get the recent value by inserting it at the head of the list and removing the oldest one from the rear of the list. As an example, let us suppose the list of recent five acceptances and

rejections is {1, 0, 1, 0, 0}, where 1 represents flow acceptance and 0 represents flow rejection. In this case the blocking probability is 3/5, and the oldest and newest values are 1 and 0 respectively. Let us also suppose that the new flow is rejected; the blocking probability will then be 4/5 as the newest rejected flow, 0, will be inserted and the oldest value, 1, will be removed from the list. The new updated list will be {0, 1, 0, 0, 0} and the blocking probability will be updated accordingly.

Although CBR shows good performance against other localized QoS routing PSR using bandwidth as the QoS metric and in terms of using the delay QoS metric (as we will see in the evaluation of our CBR (delay) algorithm), the criteria used for path selection in CBR, which relies on a crediting scheme, does not directly reflect on the quality of a path. Logically, the quality of a path should be measured directly based on the required QoS metric, like bandwidth or delay. This is why the second localized algorithm is proposed.

6.2.2 Delay Based Routing

The second new localized QoS routing algorithm proposed is the delay-based routing (DBR) scheme, which relies on the average delay in the path in order to take routing decisions. Unlike PSR and CBR, whether in bandwidth metric or delay metric (which performs routing decisions based on flow statistics of path blocking probability), DBR uses a completely different scheme of path selection based on collecting statistics about the actual delay in each candidate path. Calculation of the average delay for each candidate path is then used to measure the quality of the path, and upon flow arrival the path with the least average delay is used to route the incoming flow. DBR keeps monitoring the delay in the network and continuously updates each path's average delay in the candidate path set. Using average delay for the path directly reflects on the actual quality of the path.

The delay considered is mean delay and thus the end-to-end delay is the sum of the delay of each node / link in the path considered since the expectation (mean) of a sum of random variables is the sum of the individual expectations. This applies whether the random variables (in this case delay) are independent or not.

When a new connection arrives at a source node the signalling process starts to select a path to route a flow. The source node computes the path that may satisfy the QoS delay requirements. It uses a setup message to travel along the selected

path with each connection request. The message stores the delay via the outgoing link, and each intermediate node performs an admission test for the outgoing link, adding the outgoing link delay to the previous delay. If the delay that the message experiences is less than the QoS delay, the delay is reserved for that flow and the message is forwarded to the next node. If the delay experienced in the selected path is less than the QoS delay, as in (3), then the flow is accepted. Otherwise, if the delay over the selected path exceeds the QoS delay, and/or the delay constraint of any existing path is exceeded a failure message is propagated back to the source node and the flow is rejected. This means that the delay over that path does not satisfy the delay constraint. The information regarding flow acceptance or rejection is acknowledged to the source node in order to collect flow statistics. It should be noted that the mean delay is not considered a QoS metric by some authors but we categorise it as such here for consistency. It is, in any case, guaranteeing an average of the QoS. The pseudo code for the DBR algorithm is as follows:

PROCEDURE DBR ( )

Initialize

Set Pavg = 0, $\forall$ P $\in$ R

DBR ( )

1. Set P=min {Pavg: P $\in$ R}
2. Route flow along path *P*

3. If sum {L.delay: L $\in P\}$ $\leq$ QoS_Delay & for all existing paths P` {L`.delay : L $\in P$` , $P$` $\cap$ $P = \phi\}$ $\leq$ (QoS_Delay)` (flow accepted)
4. Calculate Average delay (P)
5. P.delay=sum {L.delay: L $\in P\}$
6. Value= (P.prevDelay + P.delay)
7. Pavg= {Value / window size}
8. Else
9. Calculate Average delay (P)
10. P.delay=sum {L.delay: L $\in P\}$
11. Value= (P.prevDelay + P.delay)
12. Pavg= {Value / window size}

END PROCEDURE

Each source-destination pair in localized delay-based QoS routing requires a predefined set of candidate paths *R*. The main characteristic associated with every path *P* in the candidate path set is the average delay. We use Pavg to store the average end-to-end delays and update its value with every connection request. Upon flow arrival, DBR selects the path with the smallest average delay (line 1) and routes the flow along the selected path. As the setup message travels to the destination it adds the outgoing link for each hop that the message passes. At the same time it performs a comparison of the links along the path with quality of service delay to ensure that this path satisfies the delay constraint (lines 2-3). It is

important to mention that each node along the selected path stores the delay of each flow using this node. Based on the stored information each node performs an admission control for each new connection request to ensure that any existing flow does not have its QoS jeopardised by the new connection. The use of call admission control means that algorithm is not necessarily scalable since this involves global information. If the flow is accepted along the selected path, the end-to-end delay is calculated along that path, and the path delay is then added to the previous (delay) values of the path and stored in the source node (lines 4-7). As a new connection arrives to the source node, the stored values are divided by the number of connections sent in order to get the actual delay of the path. It should be noted that storing the end-to-end delay along the selected path reflects on the actual delay that the path can support. In contrast, if the flow is rejected the delay calculated so far, which is larger than the delay constraint, is added to the overall path delay and the new average is calculated as previously (lines 9-12). So, when the path's delay increases, its probability to be chosen decreases for new connections. Increasing or decreasing the path delay reflects on the actual path state and the quality of the path can be measured accurately.

Unlike CBR, which monitors flow blocking probabilities, DBR monitors delay of a path and the source node stores delay values for the accepted or rejected flow of each path. It calculates the average delay using a simple moving average (sliding window) over a predetermined period. DBR uses a fixed size of sliding window of

size W connection requests, which moves to get the most recent value by inserting it at the head of the list and removing the oldest one from the rear of the list. So, for the sliding window, the average delay will be calculated using the most recent W connection requests. For example, if {10, 15, 8, 13, 9}, represent the last five delays collected over a period W=5 for path P, the average delay that path P could support would be (10+15+8+13+9)/5=11, and the oldest and newest values are 10 and 9 respectively. Let us also suppose that the new arrival is rejected and the end-to-end delay of the path was 16. The set will be changed to {15, 8, 13, 9, 16} and the new average delay will be (15+8+13+ 9+16)/5=12.2.

6.3 Performance evaluation

This section evaluates the performance of the proposed localized algorithms: the credit-based routing scheme (CBR) and delay-based routing scheme (DBR). A global routing algorithm (Dijkstra) was also used in the comparison, since many contemporary routing algorithms, such as OSPF, are based on this. Dijkstra's algorithm finds the shortest path in terms of delay that satisfies the quality of service delay constraint. The shortest-widest path (SWP) algorithm was also in the comparison. SWP finds the paths with most available bandwidth. If there is more than one path with the same available bandwidth, the one with the least delay is selected (we use the notation Dijkstra(x) and SWP(x) to refer to these algorithms with update intervals of x time units for link state information).

6.3.1 Simulation model

We implemented our localized QoS routing schemes (DBR and CBR) based on the discrete-event simulator OMNeT++ [88, 89] , and conducted extensive simulations to test their performance. Using one of the predetermined algorithms (DBR, CBR, Dijkstra's and SWP), the simulation performs path selection, resource reservation, and admission control at flow level.

Due to the varying performance of algorithms with underlying network topologies, we also used different types of network topologies [93]. We used an ISP topology in the simulation, which is widely used in different QoS routing algorithm studies [93] [94]. 49-node Torus and Lattice regular topologies were also used in the simulation to be able to select different path lengths between each source-destination pair. Random topologies were created using C++ and OMNeT++, where the connection between any two nodes is determined by a probability using the Doar-Leslie Model [85]. Table 6.1 lists the most important characteristics of the topologies used in the experiments.

Topology	Nodes	Links	Node degree	Avg. path length
RANDOM80	80	484	6.07	2.92
RANDOM60	60	326	5.43	2.56
RANDOM40	40	156	3.90	2.77
ISP	32	108	3.37	3.17
Torus	49	196	4	3.50
Lattice	49	168	3.42	4.66

Table 6.1: Network topologies and their characteristics

6.3.2 Traffic generation

In each experiment in the simulation the network topology remains fixed. The traffic is generated by having the source node choose the destination node from amongst all nodes, except the source node, with uniform probability. Flow inter-arrival times at the source node are exponentially distributed with the mean $1/\lambda$. The mean of the flow holding time is $1/\mu$, with an exponential distribution, while the QoS delay is varied, ranging between 10 and 30 time units. We also assume that all links in the topologies are bidirectional and the mean delay time for each link is also exponentially distributed.

The parameters used in the simulation for CBR are MAX_CREDITS=5 and Φ=1. Blocking probabilities are calculated based on the most recent 20 flows for DBR

and CBR. Candidate paths between each source-destination pair in a network topology are chosen, so we include minimum hop and (minimum hop) +1, +2 in the set to get the set of candidate paths between each pair. All experimental results collected are based on at least 2,000,000 connection requests (arrivals) and the results are collected after 200,000 connections requests. Each experiment was repeated 20 times with confidence interval at 95% confidence level and found that most of the confidence intervals were not visible on the figures.

6.3.3 Performance metrics

Flow blocking probability was used to measure the performance of the algorithms. Flows are rejected when the path does not satisfy the delay constraint. The blocking probability is thus defined as:

$$\text{Flow blocking probability} = \frac{\text{No of rejected requests}}{\text{No of requests arriving}}$$

In our fairness calculation, the well-known Jain's fairness index [91] was used to evaluate the fairness of the algorithms.

6.3.4 Simulation results

6.3.5 Delay constraints

It is acknowledged that blocking probability is greatly affected by the tightness of the delay QoS constraints. So if the values of the required delay constraints are very large then most of the paths in the network will satisfy the constraint since it is easy to find a path that satisfies the large delay constraints. In contrast, if the values of the required delay constraints are very small then most of the connection requests will be blocked even if we search the whole network.

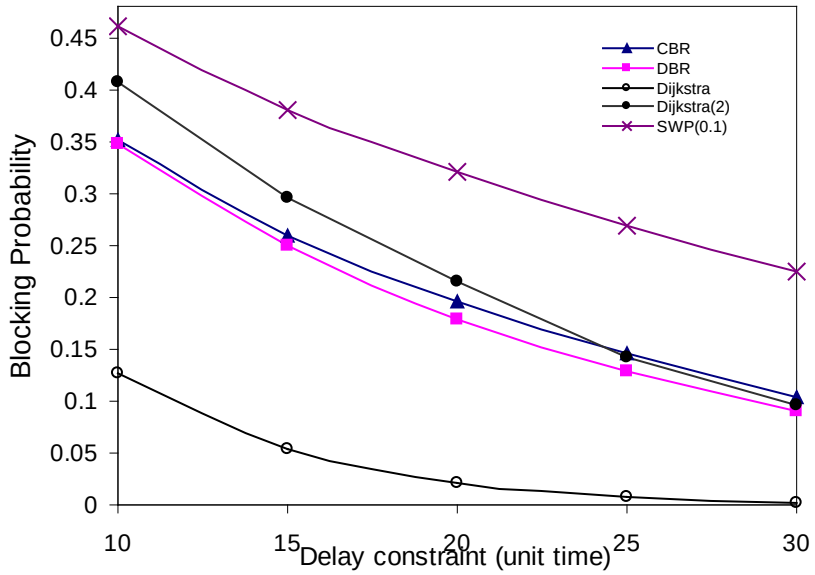
Figure 6.1 shows flow blocking probability plotted against different ranges of delay constraints for different types of network topologies. The characteristics of these topologies were described earlier in Table 6.1. It can be noted that all algorithms satisfy most flows under large delay constraints (constraint 30), which can be expected as the probability of finding a path that satisfies a large delay constraint is high and most flows will be accepted. However, performance under some constraints may be significantly degraded based on the average path length in the topology. This can be noticed in Figure 6.1 (d), where the Lattice topology gives poor performance compared with the other topology under a delay constraint of 30 due to the fact that it has the largest average path length in number of hops (averaged across all source-destination pairs). Whereas in Figure 6.1 (e), all algorithms give superb performance with the same constraint for

Random 60, which has the smallest average path length. The blocking probability increases gradually as the constraint tightens over incoming flows, which implies that flows with small delay requirements are hard to route, as expected.

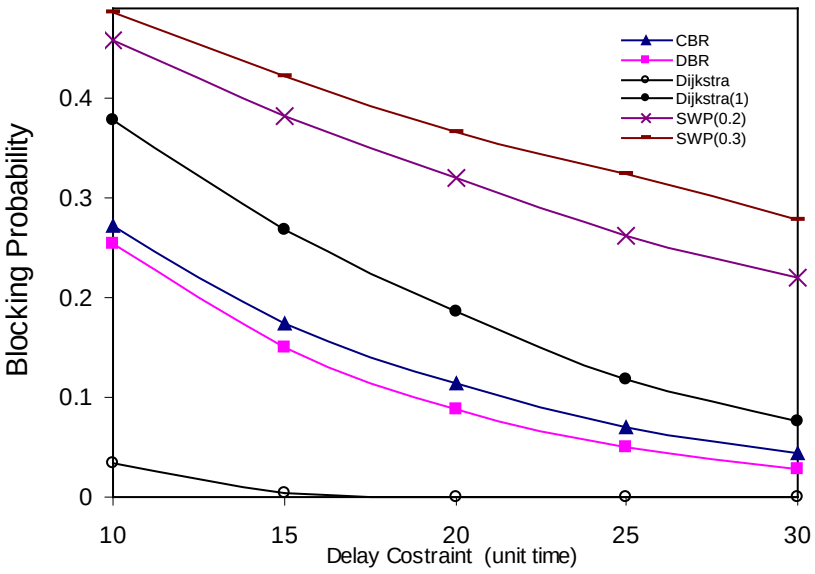
The performance of the SWP and Dijkstra algorithms is significantly affected by the update interval. We can see that as the update interval of the global state information increases, its performance degrades significantly and its blocking probability increases rapidly. This is due to the path selection of Dijkstra and SWP algorithms, which is based on the periodic update of QoS global state information that does not respond quickly to the change in network state and sticks with the current feasible path until the next update interval becomes available. It should also be noted that SWP gives the worst performance, even with a small update interval (0.1). This is likely due to the fact that the SWP algorithm finds the paths with highest amount of bandwidth and these may correspond to longer paths. Flows routed via longer paths are likely to experience larger delays which will increase the blocking probability.

In the case of localized routing algorithms CBR and DBR, we can notice that both algorithms perform well in all network topologies. Their blocking probabilities increase gradually as the delay constraint increases; which is not the case in Dijkstra's algorithm, where they increase sharply under the same constraint. This is due to the effect of alternative routing, which does not rely on global state information for path selection as in Dijkstra's algorithm. CBR selects the path

with the maximum credits as long as it does not reject flows, since credits of the selected path are changeable according to blocking probability. This leads the CBR to select alternative paths with the updated credit. However, flow rejection will cause the choosing of an alternative path with more credits. In the same way, DBR selects paths with the least average end-to-end delay, which gives more scope to select paths as long as they satisfy QoS delay. On the other hand, the DBR mechanism avoids the crediting scheme associated with the CBR scheme by selecting the path based on its quality satisfying QoS delay. The path selection method used in DBR performs well under varying delay constraints and network topologies when compared to CBR and Dijkstra's algorithm with small update intervals, as shown in Figure 6.1.

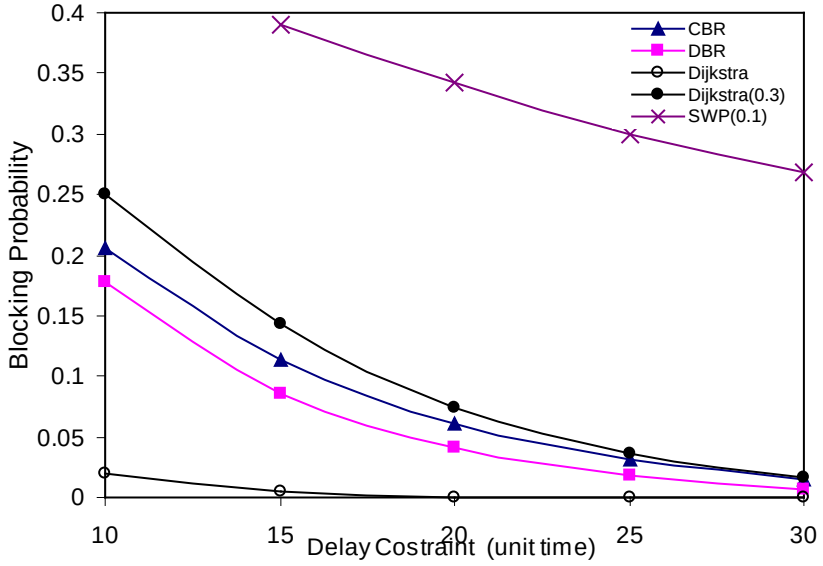


(a) ISP topology

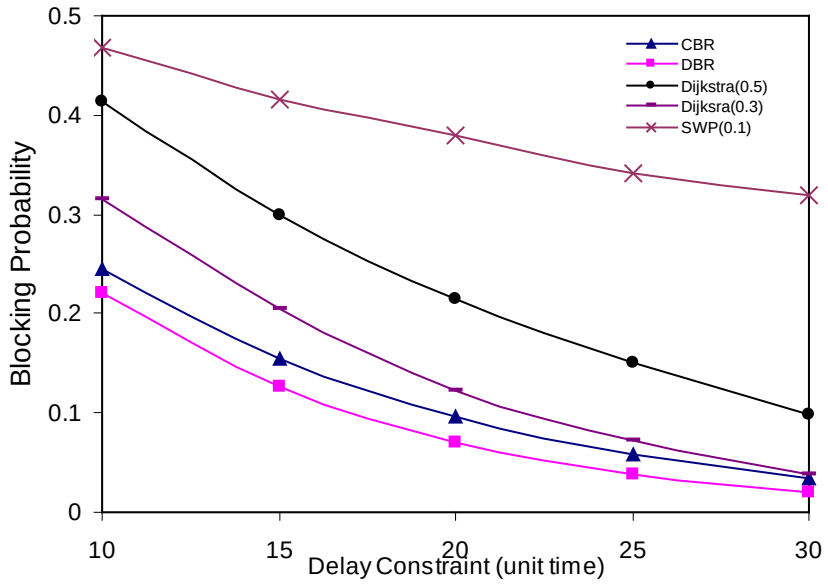


(b) Random 40 topology

Figure 6.1 Delay constraints in different network topologies



(e) Random 60 topology



(f) Random 80 topology

(d) Lattice topology

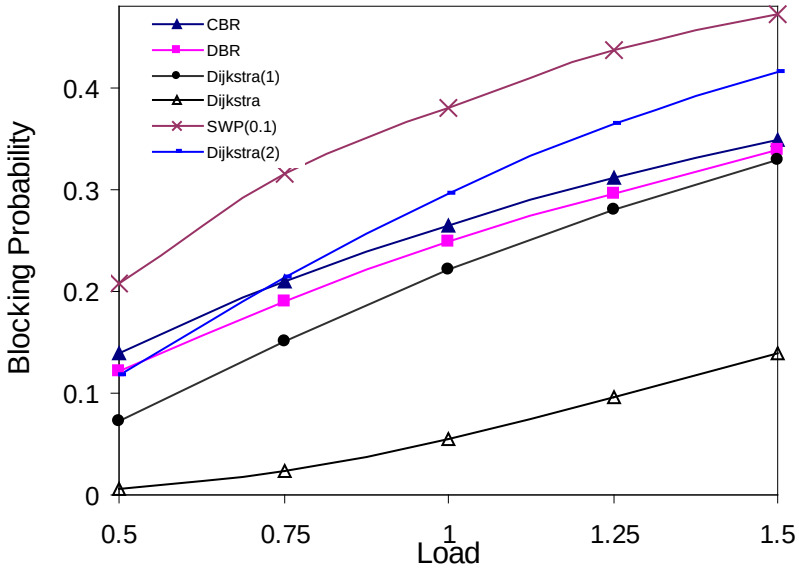
Figure 6.1 Delay constraints in different network topologies (Continued)

Figure 6.1 Delay constraints in different network topologies (Continued)

6.3.6 Impact of network topologies and varying arrival rates

In Figure 6.2 the flow blocking probability is plotted against different ranges of arrival rate for different types of network topologies. We start by offering a small arrival rate, 0.5, whereby all algorithms can accept most of the arrivals as long as the average path length is small. We then increase the rate to see how the local and global algorithms perform. From this Figure we can observe that in Random 60 and Random 80, CBR and DBR schemes give superb performance compared with Dijkstra and SWP algorithms, even with a small interval update (0.3), (0.1) of global state information. However, because of path selection in the DBR algorithm, which relies on the end-to-end delay on a path, its blocking probability is better than CBR, again in all topologies. When the arrival rate increases, CBR and DBR adapt to the change and maintain their relative performance. This can be seen in random topologies, as the blocking probability increases gradually, whereas Dijkstra's algorithm can't react promptly to changes in arrival rates and performs poorly as the arrival rate increases. This can be expected as the periodic updates do not respond quickly enough to rapid variations in flow arrival. Similarly, SWP gives poor performance for all topologies; this is expected due to the behaviour of the path selection in the algorithm, which utilizes longer paths that have the widest bandwidths. Therefore, it consumes more resources and may

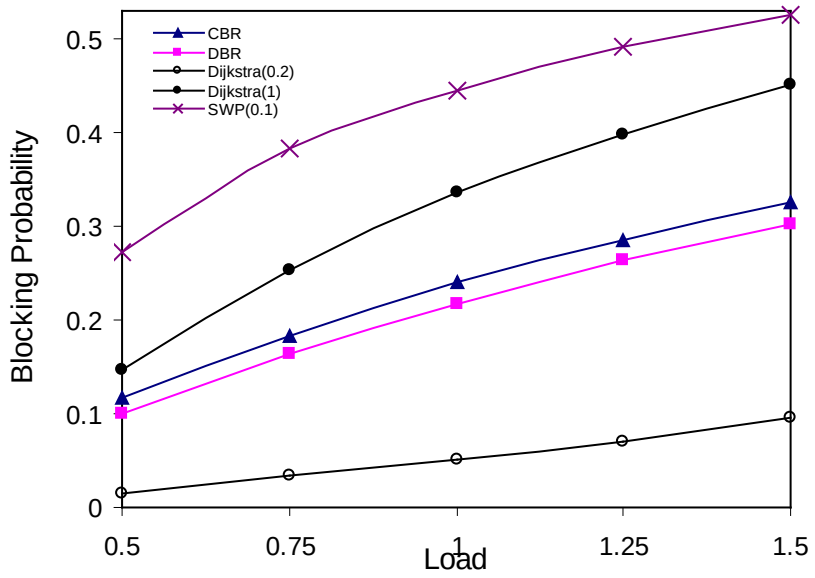
block more flows with large delay constraints. In the case of the ISP and Torus topologies, DBR and CBR fail to perform better than Dijkstra (1) in the ISP topology and Dijkstra (0.2) in the Torus topology, but still give good results against Dijkstra (2) and SWP (0.1) in the ISP and Dijkstra(1) in Torus. This is most likely because the Torus is a regular topology and there would be less



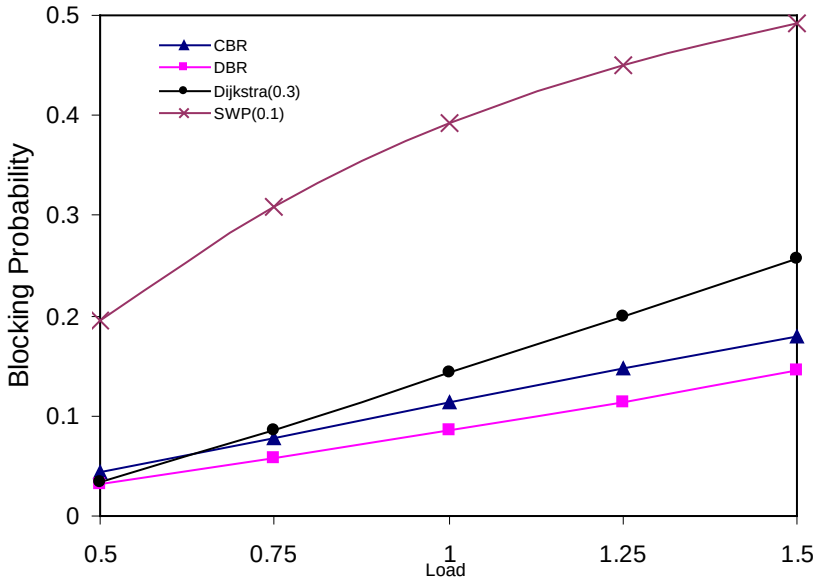
(a) ISP topology

Figure 6.2 Impact of varying arrival rates in network topologies

likelihood of route flapping with Dijkstra's algorithm.

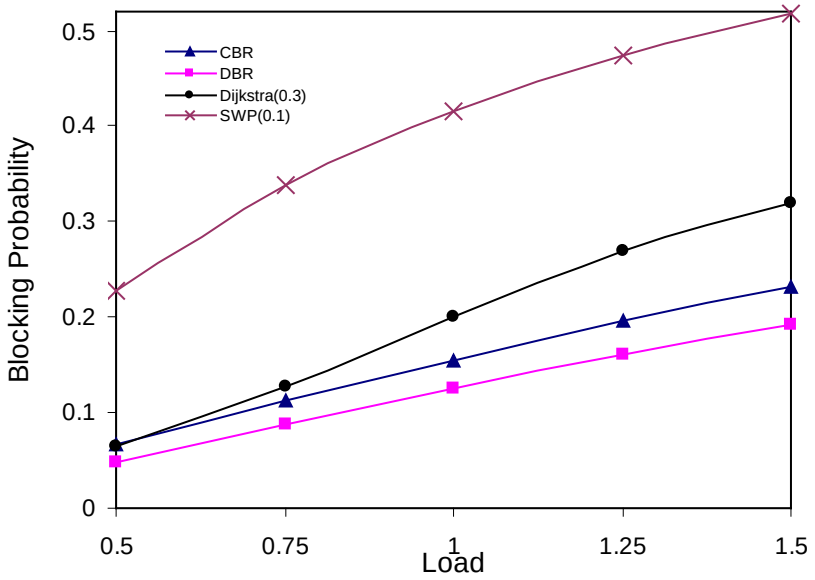


(b) Torus topology



(c) Random 60 topology

Figure 6.2 Impact of varying arrival rates in network topologies (Continued)



(f) Random 80 topology

Figure 6.2 Impact of varying arrival rates in network topologies (Continued)

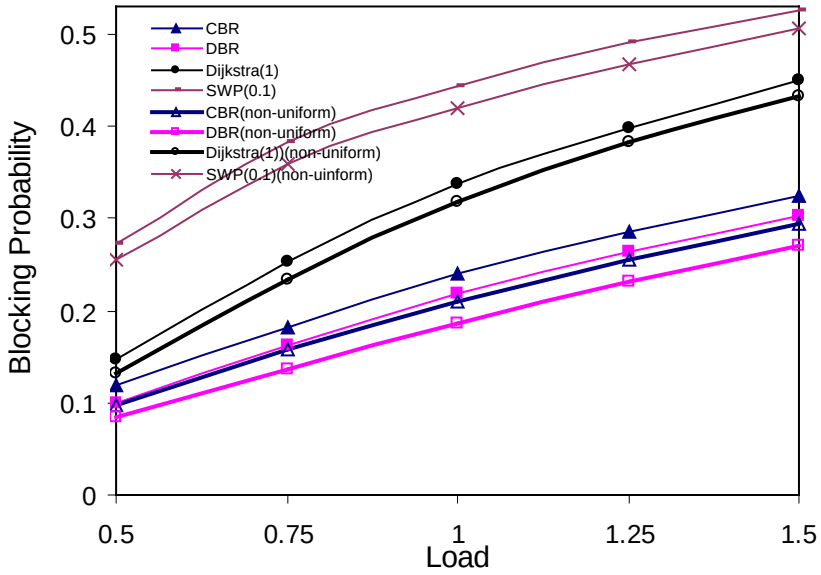
6.3.7 Impact of varying non-uniform traffic

So far the destination nodes have been chosen from a uniform random distribution. However, some source nodes may receive more flows to specific destination nodes, or in the case of the communications in specific subnets, which are usually higher than the communications across subnets. It is also emphasized that the uniform end-to-end IP QoS solution is not realistic [97]. For these reasons, the Torus and the Random 60 topologies have been virtually divided into

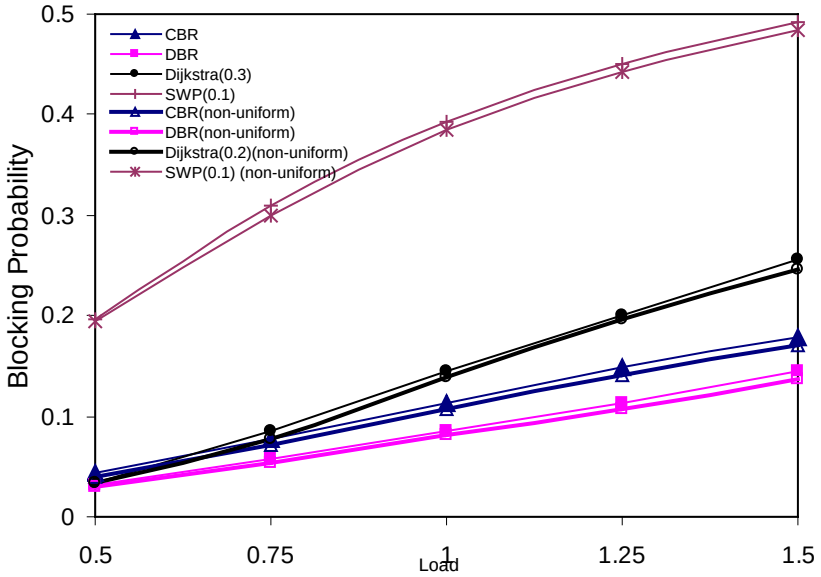
two subnets[1], so flows routed in the same subnets are three times more frequent than flows routed across subnets.

In Figure 6.3 flow blocking probability is plotted against different ranges of delay constraints using uniform and non-uniform traffic for different Torus and Random 60 topologies. It can be noticed that under non-uniform traffic all algorithms perform well in random and regular topologies compared to uniform traffic. This is due to the fact that a source node needs to have more up-to-date state information for frequent destinations; so in the case of Dijkstra and SWP algorithms, more QoS updates need to be exchanged, which increases the overhead in the network. However, since a source node in the localized QoS routing mechanism collects statistics about network state, it does have more accurate information about frequently used destination nodes. These specific results illustrate the ability of localized QoS routing algorithms to perform better than global QoS routing algorithms in terms of non-uniform traffic without increased overhead in the network, although this cannot be considered a general conclusion.

1 In fact, we have divided the topologies into three subnets and found that the results are practically identical (not shown in the graphs).



(a) Torus topology

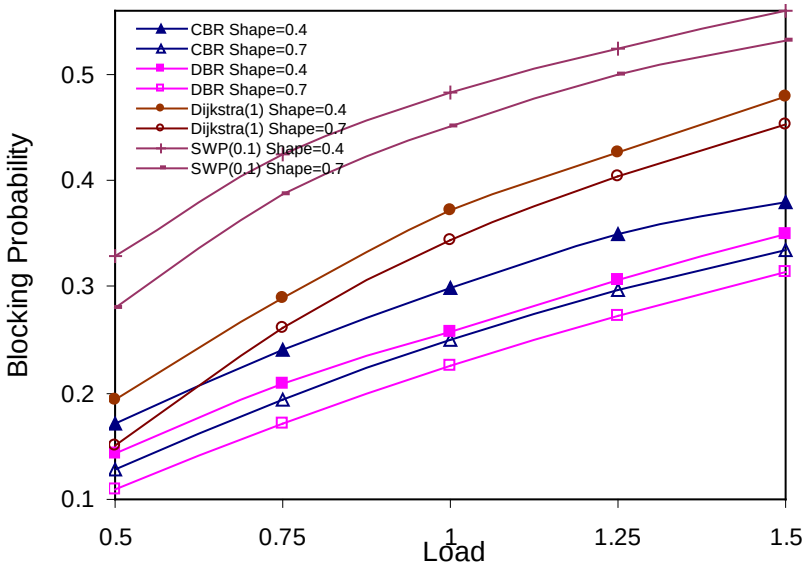


(b) Random 60 topology

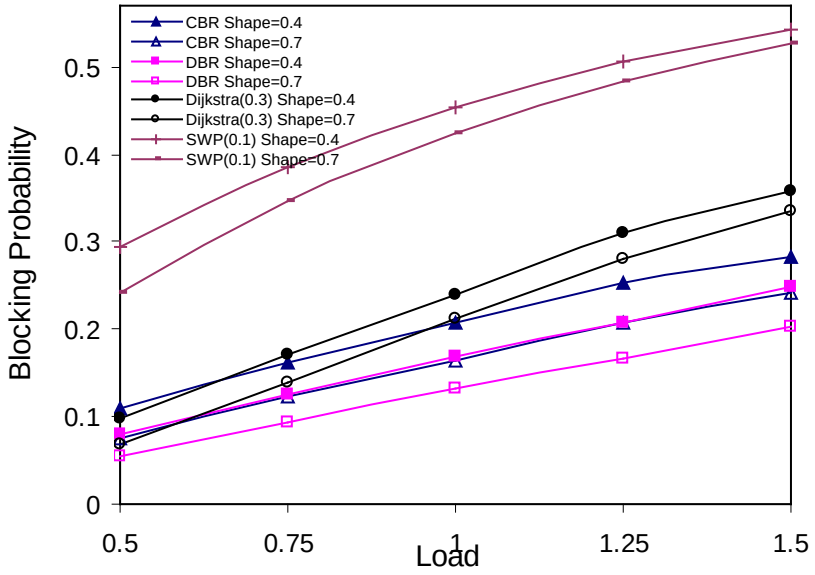
Figure 6.3 Impact of varying non-uniform traffic

6.3.8 Impact of bursty traffic

We model bursty traffic in the Torus and Random 60 topologies to study the effect of bursty connection arrivals on localized and global routing algorithms. Following [1] [96], the burstiness of traffic is modelled using a Weibull distribution with two different values of shape parameter of the distribution, 0.4 and 0.7; where burstiness is increased with a smaller shape value. Figure 6.4 shows the blocking probability plotted against different ranges of arrival rate, from 0.5 to 1.5, with two shape values for each algorithm. As can be noticed, the blocking probability for Dijkstra and SWP algorithm in Figure 6.4 (bursty) for Random 60 and Torus topologies with update intervals of 0.3 and 1 respectively is significantly higher than that in Figure 6.2 (Poisson). This can be expected because bursty connection arrivals increase blocking by making it harder for the source node to find feasible paths compared to Poisson traffic, even for small update intervals.



(a) Torus topology



(b) Random 80 topology

Figure 6.4 Impact of Bursty traffic

In localized QoS routing algorithms the blocking probability for bursty traffic also increases compared to non bursty traffic, but not as high as global routing algorithms. This is because a source node in a localized routing algorithm makes

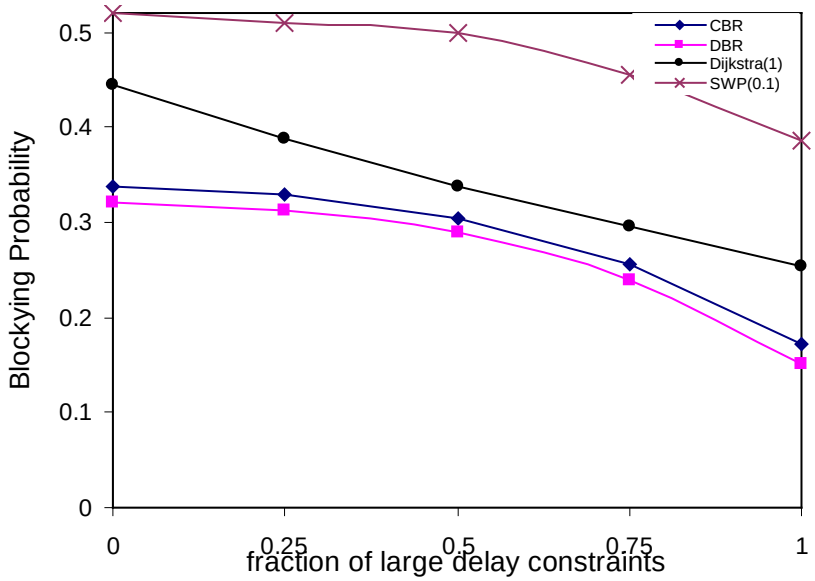
its routing decisions based on its view of the network, not on the global state information. However, DBR gives superior performance under bursty traffic with shape 0.7 compared to CBR; it even performs as non-bursty traffic in both topologies. Good performance is also found with shape 0.4, which is burstier, with almost the same CBR blocking probability in Random 80 with shape 0.7, which is less bursty. This is obvious, since CBR takes routing decisions based on blocking probability, which increases with burstiness; whereas DBR takes routing decisions based on average end-to-end delay in a path. The path selection process in DBR, which is based directly on the required QoS metric (end-to-end delay) reflects on the quality of the path.

6.3.9 Impact of heterogeneous delay constraints

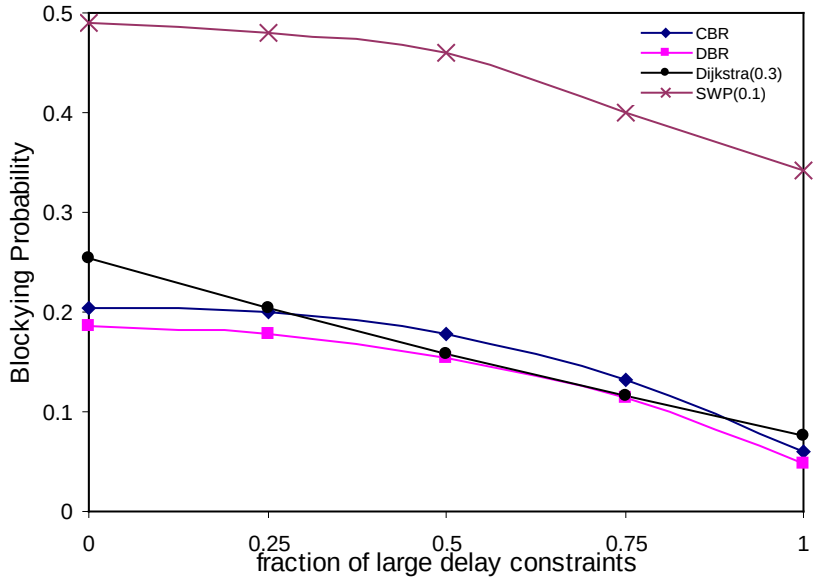
The experiments so far have seen only one delay constraint being used as QoS delay. In the following we study the effect of using more than one delay constraint on Dijkstra, SWP, CBR, and DBR. We consider two delay constraints in order to study the impact of large and small constraint flows, but having the same arrival time and holding time. The value of the delay constraints for both types is 20 for the large constraints and 10 for the small constraints, with a mean inter-arrival rate of 1. The holding times for all flows are exponentially distributed with a mean of 2.5. Performance is measured by mixing fractions of small and large delay constraints, keeping the inter-arrival mean time fixed at 1.

Figure 6.5 shows flow blocking probability plotted against the fraction of large delay constraints. It can be noticed that the blocking probability of the three algorithms decreases as the fraction of large delays increases in random and regular topologies, which is expected since it is easier for a source node to find a feasible path with a large delay constraint. Dijkstra and SWP give poor performance, regardless of the fraction of small delay; with better performance in Dijkstra's algorithm since it directly finds the least delay. Whereas SWP selects the least delay among widest paths which usually have more hop counts than the shortest paths and thus have difficulty satisfying small delay constraints.

On the other hand DBR gives the best performance, which is expected, since DBR continuously monitors the end-to-end delay in each path and the requested QoS delay is known before path selection for both large and small delay constraints. CBR lies in the middle and selects the path with the maximum credits, as long as it does not reject flows, since credits of the selected path are updated after every flow routed along the path. However, any flow rejection will cause its credits to decrease and an alternative path with more credits to be selected. It can also be noticed that the difference in performance remains fixed between CBR and DBR, and they have good performance. Dijkstra's algorithm for Random 60 has 0.3 update intervals, which is a very small update interval.



(a)Torus topology



(b)Random 60 topology

Figure 6.5 Impact of heterogeneous delay constraint

6.3.10 DBR and CBR stability and sensitivity to their parameters

It is important for good routing algorithms to be able to stabilize with rapid changes in network state, since out-of-date information in global routing algorithms collected about network state may cause route flapping with a large update interval [98].

It should be noted that localized QoS routing algorithms does not suffer from what we call the synchronization problem unlike global QoS routing algorithms. There are several reasons for the stability of localized approaches: 1) The traffic is distributed among few candidate paths instead of finding the best path based on inaccurate information; 2) The source node only is acknowledged with information about flow that the node sends and not from all network nodes and hence less variation. Because of the way the information is distributed among the candidate paths in the localized approach and the behaviour in which network resources are used in a beneficial manner the localized approach is more stable. It is also should be stated that the inaccurate information in the localized approach is not compared with inaccuracy in global information since source nodes collect information generated from itself and not information distributed periodically which soon becomes out of date. Moreover when the network is in a stable state the information acknowledged to source nodes would not become outdated and consequently each source node would have a reasonably accurate view of the network. However in case of lost information the localized approach source nodes

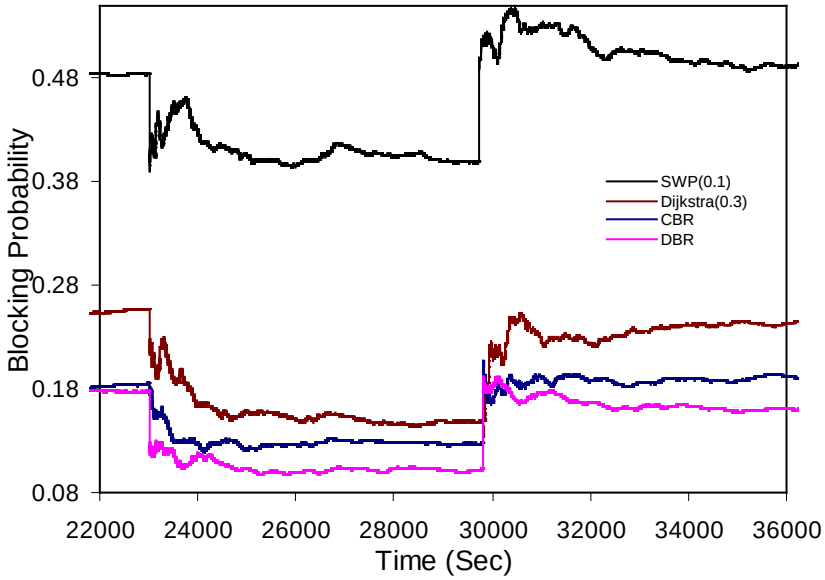
can update information, as in case of rejected flows, and the localized approach will adapt quickly to the change in network state as we can see in Figure 6.6 (a).

The blocking probability under Dijkstra (0.3), SWP (0.1), CBR and DBR as a function of time is plotted in Figure 6.6 (a). We consider three levels of connection arrivals offered to the Random 60 topologies; where arrival of mean 1.5 is offered to the network and then step decreased to 0.75 and then step increased to 1.5. It can be seen that under all these arrivals, CBR and DBR adapt quickly to rapid changes in arrivals and their blocking probability stabilizes after a short time of fluctuations when compared to Dijkstra and SWP. This is due to the fact that each source node performs routing based on its local view of the network state. So we can claim that unlike global routing algorithms, which exhibit route flapping due to exchanging global information, localized QoS routing algorithms are more stable than global QoS routing algorithms and do not exhibit such behaviour.

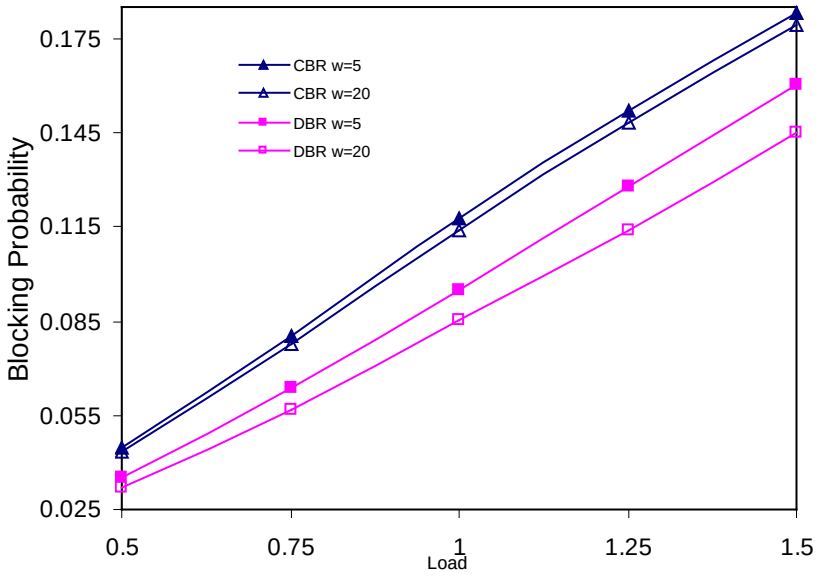
Since DBR monitors the end-to-end delay in each path of the candidate path sets, and CBR continuously records the credits associated with each path as flows are accepted or rejected along the path, a sliding window W with a predetermined period is used to record delay, and 1 and 0 upon flow acceptance or rejection for DBR and CBR, respectively. In Figure 6.6 (b), flow blocking probability is plotted against inter-arrivals using the two values of W connection requests, 5 and 20. It was found that the blocking probability decreases slightly as the value of W

increases in CBR, whereas the blocking probability of DBR decreases significantly as the window size increase. This parameter controls the observation period of the path delay and credit; so a longer period is better to get a good estimation of how good or bad the path is. Figure 6.6 (c) shows blocking probability plotted against different values of window size for Random 40, with the fixed inter-arrival time mean 1. We have found that setting W to different values has an insignificant impact on blocking probability.

CBR uses Φ to control the usage of alternative paths and MAX_CREDITS to determine the maximum credits for each candidate path. In Figure 6.6 (d), flow blocking probability is plotted against MAX_CREDITS using values between 0.1 and 5 for MAX_CREDITS. It can be seen that CBR is not sensitive to MAX_CREDITS values unless small. Choosing small values of MAX_CREDITS may not give good measurements of path quality as it will reach zero after a small number of flow rejections. It has also been shown in Figure 6.6 (e) that CBR is not very sensitive to the choice of Φ where flow blocking probability is plotted against load using the values 1, 0.9 and 0.8 in the Random 80 topology. As we can see 0.9 gives better performance than 1, and this is because 0.9 gives more priority for a minhop path to be used, which is preferred by the delay metric.

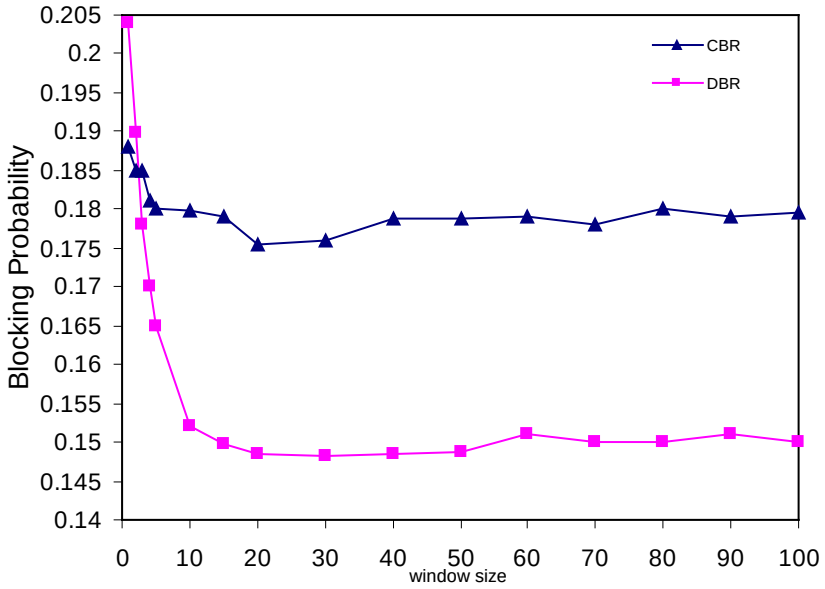


(a) Response to rapid arrival variation in Random 80

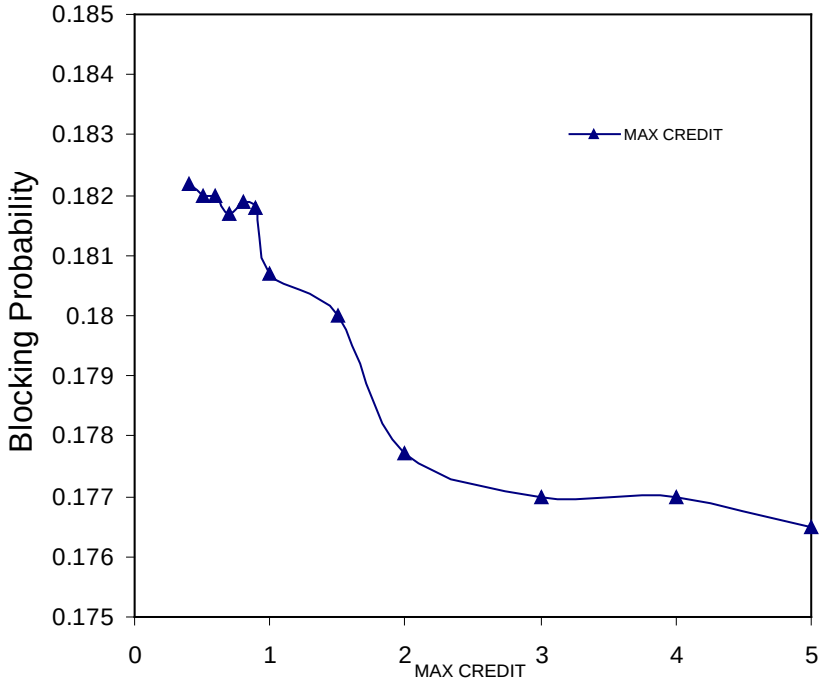


(b) Choices of W in Random 60

Figure 6.6 CBR and DBR Stability and sensitivity to their parameters

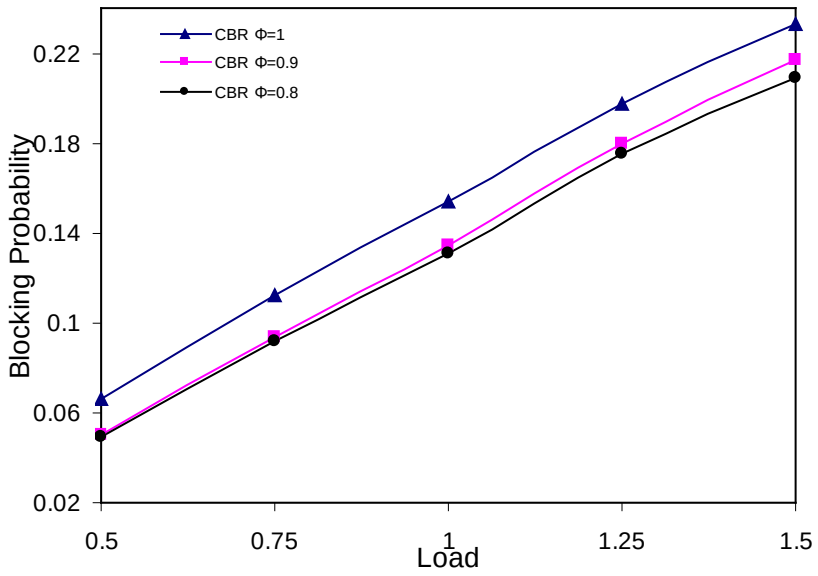


(c) Window Size (connection requests) in Random 40



(d) Choice of MAX_CREDIT for CBR in Random 40

Figure 6.6 CBR and DBR Stability and sensitivity to their parameters (Continued)



(e) Choice of parameter Φ for CBR in Random 80

Figure 6.6 CBR and DBR Stability and sensitivity to their parameters (Continued)

6.3.11 Load Balancing

Figure 6.7 compares the average Jain's index of DBR, CBR and Dijkstra (1) in the Random 80 topology. The values for the index reflect an average taken at regular intervals of time over 2,000,000 connection requests since the delay fluctuates rapidly over time. The average Jain's index is plotted as a function of offered load. From the figures, we see that Dijkstra gives the least fairness, which is expected since Dijkstra always tries to select the path with least delay. So it keeps

using the same path between each source and destination nodes until the next update interval indicates the overloading of that path and causes unbalanced load distribution in the network. In the case of the SWP algorithm, of which its main feature is load balancing, the index is better than Dijkstra's algorithm. This is expected since SWP distributes the load based on the highest available bandwidth and this would give more scope for different paths to be selected.

However, CBR gives a better fairness index than Dijkstra and SWP algorithms since it selects a path with maximum credits. As the load increases, we can notice that the fairness is increased, which is expected since any flow rejection will decrease the path credit and an alternative path will be selected. Unlike CBR, which sticks with the same path as long as this path has the most credits, DBR distributes the load among the candidate paths. This can be noticed as it gives the best fairness index. DBR selects the path based on its delay and may select the alternative without flow rejection. As a result of its path selection mechanism, it gives a good fairness index regardless of the load in the network.

It is interesting to note that the values of Jain's index in Figure 6.7 appear much lower than those obtained when bandwidth is used as the single QoS metric [99]. This is likely because delay is a path based metric whereas bandwidth is a link based metric. With link based metrics any paths that include bottleneck links are dropped early, whereas with path based metrics, such as delay, bottleneck links can exist within a path for a long period provided that the other links in the path

have low delays. The end-to-end delay on the path may therefore still satisfy the QoS but the load on the links may be very unbalanced.

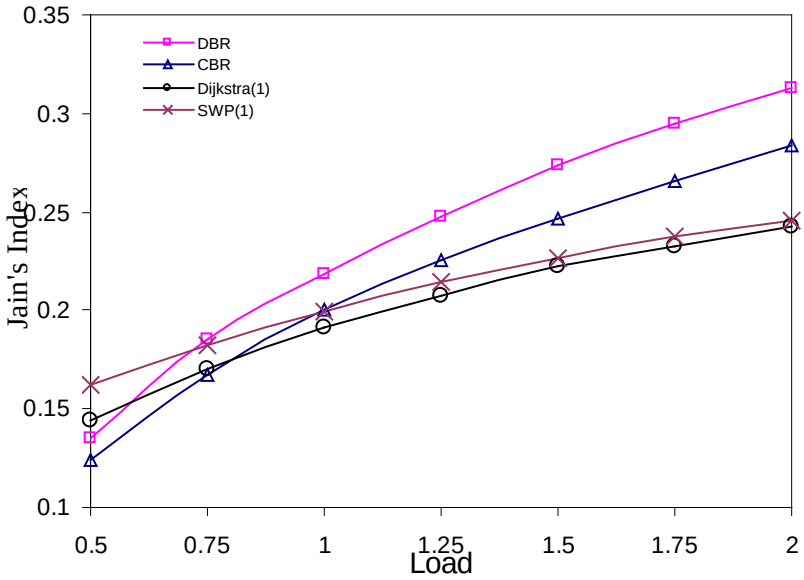


Figure 6.7 Average Jain's index in Random 80

6.3.12 DBR and CBR time complexity

Global QoS routing algorithms performing a variant of Dijkstra's algorithm to find the shortest path or widest path, or both, like SWP, take at least O (N log $N+L$) time; where N is the number of nodes and L is the number of links in the network. On the other hand, the complexity of selecting a path among the set of candidate paths R in CBR and DBR is O (|R|). CBR needs to update blocking probabilities, which takes a constant time O (1). Similarly, DBR needs to update the average delay, which also takes O (1).

6.4 Summary

In this chapter, we have developed two localized QoS routing algorithms: CBR, which uses a simple crediting scheme that is increased upon flow acceptance and decreased upon flow rejection; and DBR, which relies on actual delay on the path in order to take routing decisions. We demonstrated through simulation that the two proposed algorithms, although simple, outperform global routing schemes under different traffic loads and network topologies, even for a small update interval of link state. Our general results suggest that:

- Localized QoS routing should be based on schemes that explicitly reflect the quality of a path, rather than schemes that are based indirectly on the path quality.
- Localized QoS routing can be applied to the internet, as it performs well in random subnets as well as across subnets.
- Localized QoS routing can be employed to routes in a network with multi-constraint delay requirements or heterogeneous traffic.
- Localized QoS routing performs well with non-uniform traffic patterns and would lend itself well to a partitioned network where the partitioning reflects the non-uniformity.

Chapter 7

Localized QoS Routing in Hierarchical Networks

7.1 Introduction

In the last two chapters we proposed localized QoS routing algorithms to enhance the scalability of QoS routing algorithms in terms of routing overhead and time complexity. However, with the growth of network size, the overhead at each node in the network increases significantly. Employing hierarchical routing [100] provides a scalable solution as an alternative to flat routing. In hierarchical routing, a network is partitioned so it is grouped into levels; the routing in the lowest levels is flat as each node maintains detailed state information about nodes in the same sub-network and aggregated state information about other sub-networks, so when forwarding a packet to another sub-network, the nodes rely on the higher level, which has more information about the other sub-networks, in order to locate the destination sub-network and from that to relay the packet to the final destination.

Hierarchical models are often used to reduce the size of the global state using aggregated global state that represents the hierarchical structure of the network. The aggregated information about a sub-network should be as accurate as possible and the size of the aggregated state should be reduced as much as possible [101]. Source routing mechanisms are used in hierarchical algorithms and the computations of feasible paths are carried over many nodes. So, the hierarchical routing mechanism combines the advantages of source and distributed routing schemes. In hierarchical routing algorithms the state information maintained at each node is logarithmic to the size of the complete global state and therefore hierarchical routing algorithms scale very well in large networks.

However hierarchical routing suffer from the inaccuracy of state information as a result of the state-aggregation, which has a significant negative impact on the routing algorithm [102] [103]. As the number of hierarchical levels increases, the inaccuracy introduced by aggregation increases too. Moreover, there is no optimum approach to represent a logical sub-network that can be single node representation, mesh representation or star representation and each of them has its pros and cons [54]. Each logical link state in the top level represents the combination of more than one lower-level link state, so it is very hard to aggregate these links and their state information into one logical link. The problem becomes more complex when dealing with multiple state information

metrics, and the aggregation problem becomes more challenging. However, some solutions can be found in [102] [58].

In this section we propose a localized bandwidth based QoS routing in two levels of a hierarchal network. The hierarchical localized bandwidth based routing scheme (HBBR) is a hierarchical source routing scheme whereby a source node relies on localized information that is collected locally about average residual bandwidth on the path in order to take routing decisions. In the HBBR scheme, each node collects statistics about traffic generated from itself to each of the other nodes in its sub-network, whereas each border node collects statistics about traffic generated from itself to its sub-network and to the other sub-networks via the backbone. The main motivation for HBBR is to provide a scalable and efficient hierarchical scheme based on localized information that does not require global information for each hierarchical level. We study other QoS routing schemes, such as the localized Credit Based Routing (CBR) proposed in [80]. We also use the global QoS routing scheme Widest Path (Dijkstra) in the lower level and the widest shortest path (WSP) algorithm proposed in [87] for the backbone in order to compare their performance with our scheme in terms of flow and bandwidth rejection probability under different network loads and system parameters. All routing algorithms in this paper use bandwidth as the only QoS routing metric and ignore the link propagation delay, since residual bandwidth can give an indication of the quality of a path in case of other metrics such as delay and jitter. However,

if such metrics are considered then this would involve modifying the routing algorithms to able to for these. For example, in the case of the delay metric, call admission control would need to be considered since accepting a flow might jeopardize the delay constraints of the existing flows.

7.2 The proposed algorithm

In this section we propose a new localized routing algorithm that targets hierarchical networks. The HBBR algorithm is a source routing algorithm, as the source nodes have the ability to select an explicit path to the destination node at the low level and the destination sub-network at the backbone (top level). The HBBR algorithm can be implemented on the internet using one of the various traffic engineering techniques such as Multiple-Protocol Label Switching (MPLS). The proposed algorithm assumes that the network performs resource reservation and sends signalling messages to set a path for new connections. In our proposed algorithm, residual bandwidth is used as the quality of service metric. The algorithm guarantees that the residual bandwidth on the links between the source and destination node can accommodate flow bandwidth.

Unlike PSR and CBR, HBBR uses a completely different scheme in terms of path selection, based on the collection of statistics about the actual residual bandwidth in each candidate path. A measure of the average residual bandwidth for each candidate path is then used to measure the quality of the path, and upon flow

arrival the path with the highest average residual bandwidth is used to route the incoming flow. HBBR keeps monitoring the residual bandwidth in the network and continuously updates each path's average residual bandwidth in the candidate path set. Using the residual bandwidth for the routing decision thus directly reflects the actual quality of the path.

7.2.1 HBBR routing operation

According to HBBR nodes are grouped into areas organized to be connected via the backbone. Based on the location of the source and destination nodes in the hierarchical network HBBR operates as follows:

- **Localized routing within an area**

When a new connection arrives at a source node and the destination node is located in the same area a flat routing is used. The source node computes the path that may be able to satisfy the QoS bandwidth requirements. The signalling process starts at the source node by sending a setup message along the selected path. The message stores the residual bandwidth over the outgoing link and compares it with links along the selected path to get the least residual bandwidth as follows:

$$Bandwidth(P) = \min(risdual\_bandwidth(i)), i < n$$

where n is the number of links along the selected path *P*.

Each intermediate node along the selected path performs an admission test for the outgoing link. If the residual bandwidth over the outgoing link is larger than the requested bandwidth, the bandwidth is reserved for that flow and the message is forwarded to the next node. The flow is accepted if all links in the selected path satisfy the QoS bandwidth. Otherwise, if any link over the selected path has less bandwidth than the requested bandwidth, a failure message is propagated back to the source node and the flow is rejected. This means that the bandwidth over that path does not satisfy the bandwidth constraint. The information about flow acceptance or rejection is acknowledged to the source node in order that flow statistics can be collected.

- **Localized inter-area routing**

If the destination node is located outside the sub-network, and since the source node only stores information about its area, it sends the message to the border node following the same procedures described earlier. A border node in a given sub-network (area) is a node attached to two or more areas. It has detailed information about its area and other areas' border nodes, but less information about nodes in other areas. The border node keeps monitoring the residual bandwidth in the backbone by continuously updating the average residual bandwidth for each candidate path to each sub-network in the network. Upon message arrival to the border node from a node in its sub-network, it selects the path that will satisfy the flow requirements to the border node in the destination

partition following the same procedures described earlier. At this stage, the information regarding flow acceptance or rejection to the border node in the destination sub-network is acknowledged to the border node in the source sub-network in order to collect flow statistics about the backbone. Finally, upon message arrival to the border node in the destination sub-network from outside its sub-network, it sends the message to the final destination node using the candidate path that may satisfy the flow requirements. The information regarding flow acceptance or rejection to the final destination node in the destination sub-network is acknowledged to the border node in the destination sub-network.

7.2.2 HBBR algorithm

The HBBR algorithm performs localized routing between networks inside the sub-networks and across the sub-networks via the backbone. It relies on data collected from the routing process about residual bandwidth in the two levels to make a reliable estimation about how good the candidate paths in the network are, and this will reflect on the performance of the routing of the algorithm. The pseudo code for the HBBR algorithm is as follows:

PROCEDURE HBBR ( )

Initialize

Set Pavg = CAPACITY, $\forall$ P $\in$ R

If ((source_node & destination_node) $\subset$ partition)

Routing (source _node, destination_node)

Else

Routing (source_node, source_border_node)

Routing (source_border_node, destination_border_node)

Routing (destination_border_node, destination_node)

Start **Routing** (node, node)

Set P=max {Pavg: P $\in$ R}

Route flow along path *P*

If {L.residual_bandwidth $\geq$ QoS_Bandwidth: L $\in$ P}

Calculate Average Residual Bandwidth (P)

P.residual_ bandwidth =min {Link.residual_ bandwidth: Link $\in P$}

Value= (value + P.residual_ bandwidth)

Pavg= {Pavg, Value}

Else

Value= {0}

Pavg= {Pavg, Value}

END Routing

END PROCEDURE

Each source-destination pair in hierarchical localized bandwidth-based QoS routing requires a predefined set of candidate paths R. The main characteristic of every path P in the candidate path set is the average residual bandwidth. We use

Pavg to store the average minimum residual bandwidth and update its value with every connection request. Upon flow arrival to the source node, and based on the destination node, HBBR routes the flow in the same source node sub-network to the destination node (lines 1, 1.a); otherwise it routes the flow to the source border node (line 2.a). Then it routes the flow from the source border node to the destination border node (line 2.b); finally it routes the flow from the destination border node to the destination node (line 2.c). In each level of the network, HBBR selects the path with the largest average residual bandwidth (line 4) and routes the flow along the selected path. As the setup message travels to the destination it performs a comparison over the links along the path with quality of service bandwidth to ensure that this path satisfies the requested bandwidth (lines 5-6). If the flow is accepted along the selected path, the residual bandwidth is calculated along that path, and the path residual bandwidth is then added to the previous (residual bandwidth) values of the path and stored in the source node (lines 7-10). As a new connection to the source node arrives, the stored values are divided by the number of connections sent in order to get the actual residual bandwidth of the path. It should be noted that storing the residual bandwidth along the selected path reflects on the actual bandwidth that the path can support. In contrast, if the flow is rejected zero value is added to the overall path residual bandwidth and the new average is calculated as previously (lines 12-13). So, when the path's residual bandwidth is increased its probability of being chosen is increased for new

connections. Increasing or decreasing the path residual bandwidth reflects on the actual path state and the quality of the path can be measured accurately.

Unlike CBR, which monitors flow blocking probabilities, HBBR monitors the residual bandwidth of a path and the source node stores residual bandwidth values for the accepted or rejected flow of each path. It calculates the average residual bandwidth using a simple moving average (sliding window) over a predetermined period. HBBR uses a fixed size of the sliding window of size *W* connection requests, which moves to get the most recent value by inserting it at the head of the list and removing the oldest one from the rear of the list. So, for the sliding window, the average residual bandwidth will be calculated using the most recent *W* connection requests. For example, if {2.2, 2, 0.9, 1.1, 3}, represents the last five residual bandwidths collected over a period W=5 for path P, the average residual bandwidth that the path P could support would be (2.2+ 2+0.9+1.1+ 3)/5=1.84, and the oldest and newest values are 2.2 and 3 respectively. On the other hand, if a new arrival is rejected then 0 will be added to the residual bandwidth of the path. Therefore the set will be changed to {2, 0.9, 1.1, 3, 0} and the new average residual bandwidth will be (2+0.9+1.1+ 3+0)/5=1.4

7.3 Performance evaluation

This section evaluates the performance of the proposed hierarchical localized algorithm (HBBR) and the hierarchical credit-based routing schemes (HCBR). In

its original form CBR was developed for flat networks. We have therefore modified this so it is applicable to hierarchical networks. Under HCBR, routing in each level follows the CBR algorithm and uses the crediting scheme to assess the quality of the path in the backbone and sub-networks. Global routing algorithms (Dijkstra) for the local area and WSP for routing between the areas (backbone) were used in the comparison, since many contemporary routing algorithms, such as OSPF, are based on these. Dijkstra's algorithm finds the widest path in terms of bandwidth that satisfies the quality of service bandwidth. WSP finds the minimum hop count path that satisfies the bandwidth constraint. If there is more than one path with the same length, the one with the maximum available bandwidth is selected (we use the notation Dijkstra(x)-WSP(x) to refer to this algorithm with update intervals of x time units for link-state information).

7.3.1 Simulation model

We have implemented HBBR using the discrete-event simulator OMNeT++, and conducted extensive simulations to test their performance. Using one of the predetermined algorithms (HBBR, HCBR and Dijkstra-WSP), the simulation performs path selection, resource reservation and admission control at flow level.

Due to the varying performance of algorithms with underlying network topologies, a 9-subnetwork lattice topology was used in the simulation to assess performance with a regular topology. Random 400 and 280 node topologies were

also created using C++ and OMNeT++, where the connection between any two nodes is determined by a probability using the Doar-Leslie Model [85].

In order to be able to measure the performance of our hierarchical routing scheme, two-level hierarchical networks have been developed. Figure 7.1 shows the random 400 nodes network has been split into 10 sub-networks (or areas), each of them having the characteristics of the random 40. We considered the 10 areas as the low level in the network. The top level of RAND400 (or the backbone of the network) was formed by connecting the 10 areas randomly. Similarly, figure 7.2 shows the random 280-node network has been split into seven sub-networks (or areas). However, the seven sub-networks are comprised of a random 80, a random 60, two random 40 and three random 20. Since the internet is compromised of different sizes of networks we also need to assess the effect of heterogeneous networks on the algorithms performance. The top level of the HETEROGENEOUS network (or the backbone of the network) was formed by connecting the seven sub-networks randomly. The LATTICE topology in figure 7.3 is composed of nine random 20 sub-networks. Table 7.1 and Table 7.2 list the most important characteristics of the low level and backbone topologies used in the experiments respectively.

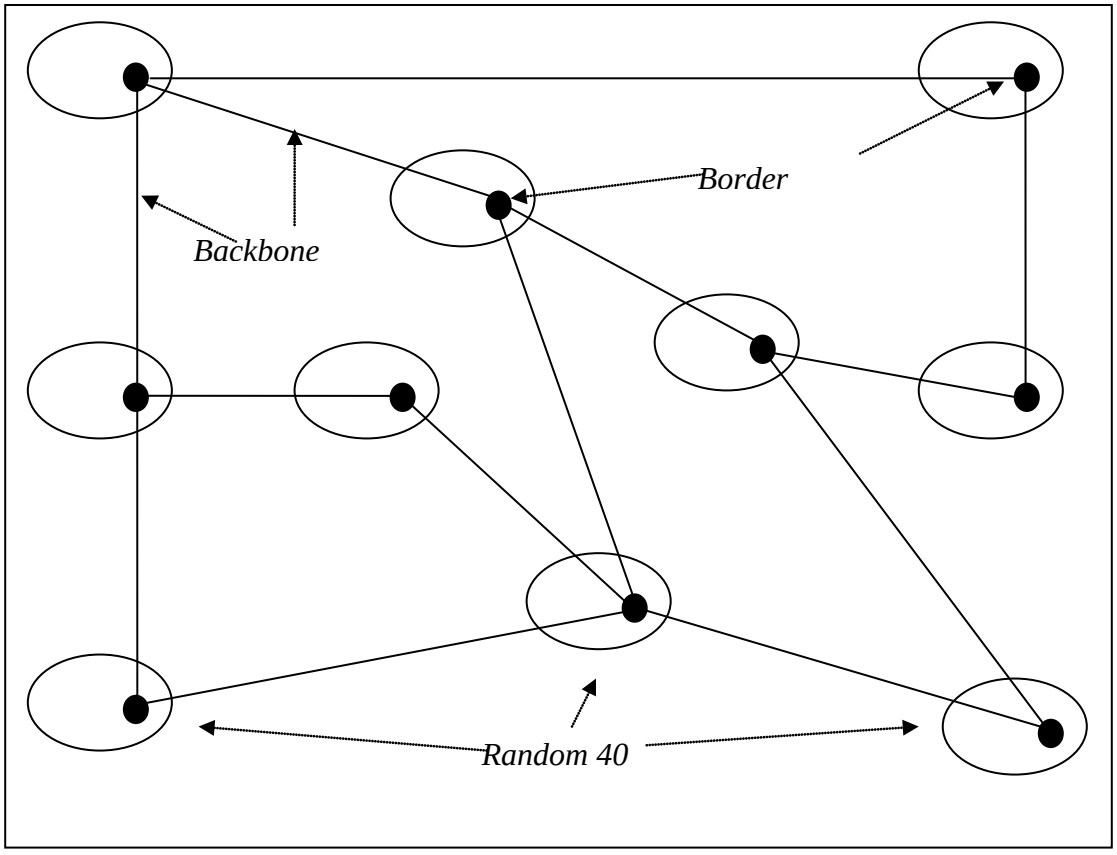


Figure 7.1 A random topology with ten subnetworks, each comprised of random 40 randomly connected nodes

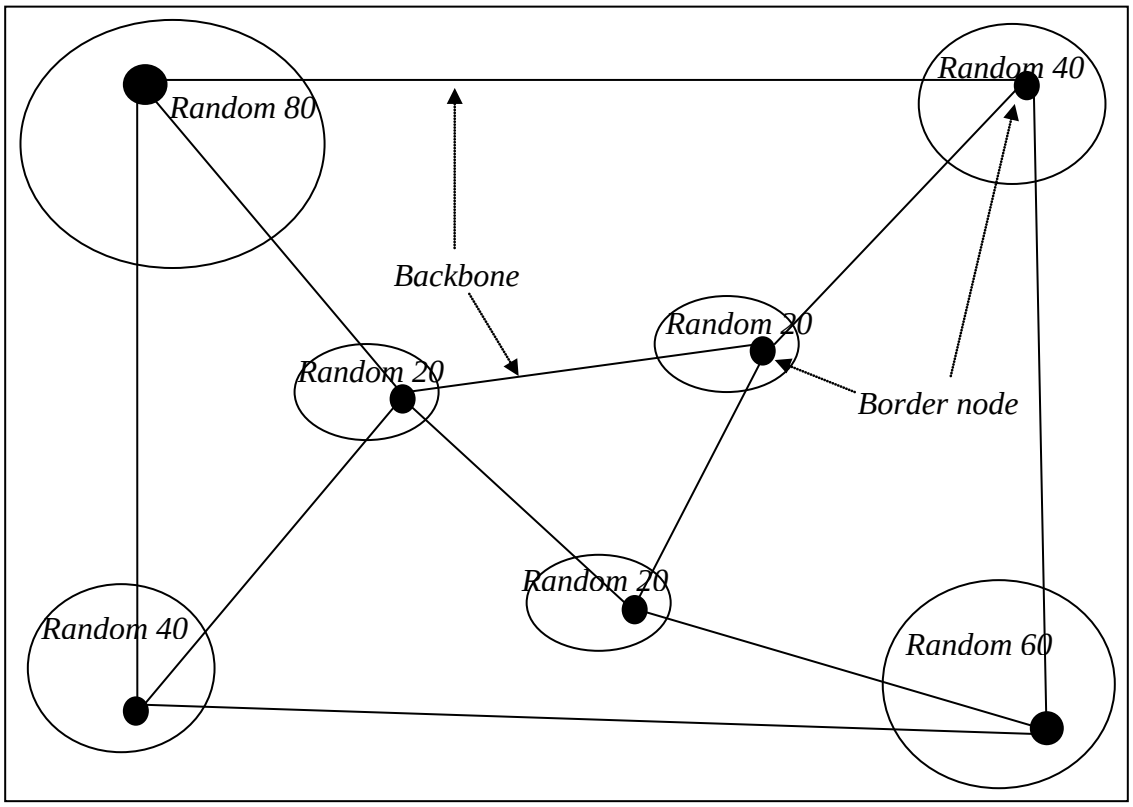


Figure 7.2 A random topology with seven heterogeneous subnetworks

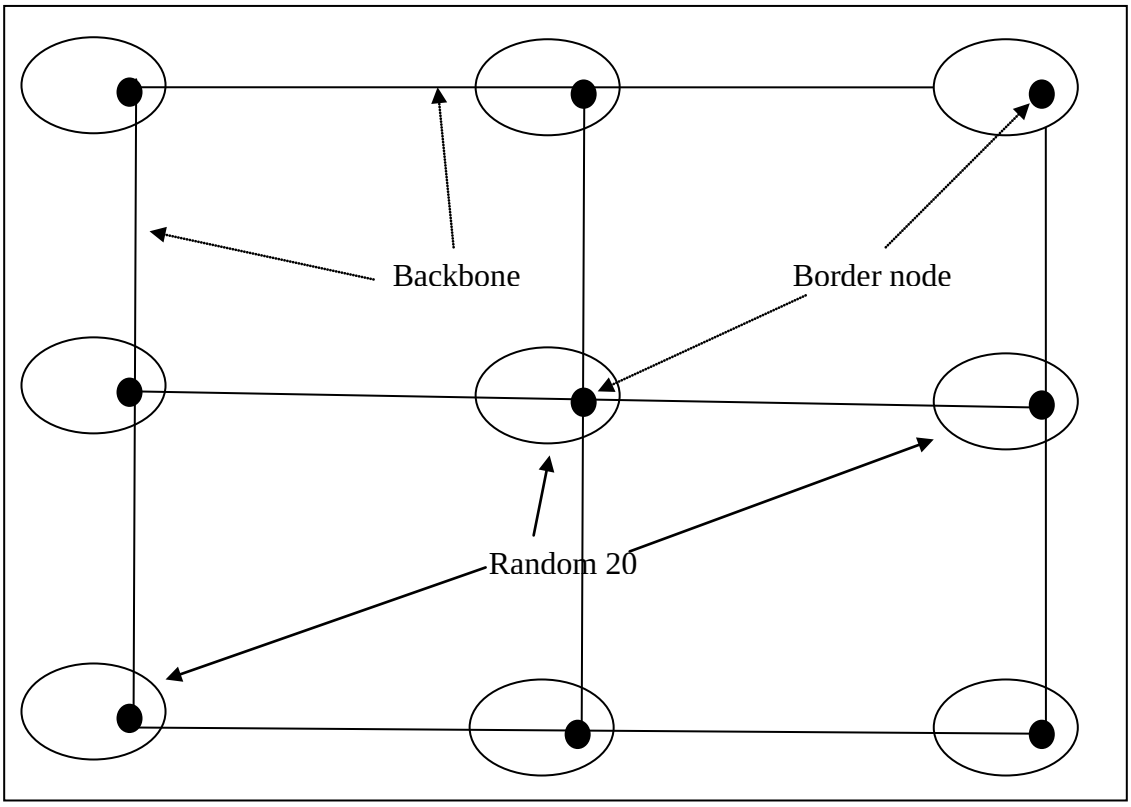


Figure 7.3 A lattice topology with nine subnetworks each comprised of 20 randomly connected nodes

Topology	Nodes	Links	Node degree	Avg. path length
RANDOM80	80	484	6.07	2.92
RANDOM60	60	326	5.43	2.56
RANDOM40	40	156	3.90	2.77
RANDOM20	20	74	3.70	2.26

Table 7.1: Sub-network topologies and their characteristics

Topology	Links	Node degree	Avg. path length
RAND400	26	2.60	2.10
HETEROGENEOUS	22	3.14	1.48
LATTICE	24	2.66	1.99

Table 7.2: Backbone topologies and their characteristics

7.3.2 Traffic generation

In each experiment in the simulation the network topology remains fixed. The traffic is generated by having the source node choose the destination node from amongst all nodes (except the source node) with uniform probability. So we used different ratios for the communication in each area against the communication between the areas. This is due to the fact that the uniform traffic for all nodes in

the whole network will generate most of the communication between the areas, and this is not necessarily the case on the internet. Flow interarrival times at the source node are exponentially distributed with the mean $1/\lambda$. The mean of the flow holding time is $1/\mu$, with an exponential distribution. The QoS requested bandwidth is uniformly distributed in the range between 0.1 and 2 MB. We also assume that all links in the topologies are bidirectional, each with the same capacity C in each direction (C=150Mbps for the areas, C=500Mbps for the backbone and links that connected areas with the backbone).

Following [1], the offered network load is $L_d = \lambda Nhb/\mu LC$, where N is the number of nodes, b is the average bandwidth required by a flow, h is the average path length (averaged across all source-destination pairs.) and L is the number of links in the network. Residual bandwidth is calculated based on the most recent 50 connection requests for HBBR. Candidate paths between each source-destination pair in each area of the low level and each source border node and destination border node for the backbone are chosen, so we include minimum hop and (minimum hop) +1 in the set to get the set of candidate paths between each pair. All experimental results collected are based on at least 2,000,000 connection requests (arrivals) and the results are collected after 200,000 connections requests. Each experiment was repeated 20 times with confidence interval at 95%

confidence level and found that most of the confidence intervals were not to be visible on the figures.

7.3.3 Performance metrics

Flow blocking probability and bandwidth rejection probability were used to measure the performance of the algorithms. Flows will be rejected when one of the links along the path from source to destination does not satisfy the requested bandwidth. The blocking probability is defined as:

$$\text{Flow blocking probability} = \frac{\text{No of rejected requests}}{\text{No of requests arriving}}$$

$$\text{Bandwidth rejection probability} = \frac{\sum_{i \in B} bandwidth(i)}{\sum_{i \in C} bandwidth(i)}$$

Here B is the set of blocked flows and C is the set of total flows. Bandwidth (i) is the requested bandwidth for path i.

In our fairness calculation, the well-known Jain's fairness index [91] was used to evaluate the fairness of the algorithms. We use Jain's index only for fairness to be consistent with pervious work on CBR [99].

7.3.4 Simulation results

7.3.5 Blocking probabilities

Figure 7.4 shows flow blocking probability plotted against the offered load in the two levels of the heterogeneous hierarchical network. The update interval of Dijkstra for the areas is set to 3, 4 (the units represent seconds) and 0. This latter reflects a situation where there is immediate knowledge of any changes (best case), which is not possible in practical terms. Two ratios of communication levels within sub-networks and between sub-networks of 2:1 and 3:1 have been used to study the effect of different loads on the network areas and the backbone. It can be noted that the more traffic there is on the backbone, the more probability there is of blocking, as the chance of finding a path that satisfies the QoS requested bandwidth across two areas in the network is low. It can also be noted that all algorithms satisfy most flows under small loads, which can be expected as the probability of finding a path that has sufficient bandwidth is high and therefore most flows will be accepted.

The performance of the hierarchical Dijkstra and WSP algorithm is significantly affected by the update interval of Dijkstra in the areas of the low level. We can see that as the update interval of the global state information increases, its performance degrades significantly and its blocking probability increases rapidly. This is due to the path selection of Dijkstra's algorithm, which is based on the

periodic update of QoS global state information that does not respond quickly to the change in the network state and sticks with the current feasible path until the next update interval becomes available. This is also the case for the WSP, and despite the link capacities in the backbone being large, the algorithm gives poor performance compared to localized schemes. However, with unrealistically up-to-date information, Dijkstra (0)-WSP (0) performs better than our scheme, regardless of the ratio of the traffic in these areas, which is as expected.

In the case of localized routing algorithms HCBR and HBBR, we can notice that both algorithms perform well regardless of the load ratio. Their blocking probabilities increase gradually as the network load increases; which is not the case in the Dijkstra-WSP algorithms where they increase sharply unless under low load. This is due to the effect of alternative routing, which does not rely on global state information for path selection as in Dijkstra's algorithm. HCBR selects the path with the maximum credits as long as it does not reject flows, since credits of the selected path are changeable according to the blocking probability. This leads the HCBR to select alternative paths with the updated credit. However, rejection of a flow will cause the choosing of an alternative path with more credits. However, HBBR selects paths with the most average residual bandwidth, which gives more scope to select paths as long as they satisfy QoS bandwidth, which reflects the path quality. On the other hand, the HBBR mechanism avoids the crediting scheme associated with the HCBR scheme by selecting the path based

on its quality satisfying QoS bandwidth. This can be noticed in figure 7.4, as the blocking probabilities of HBBR outperform Dijkstra (3)-WSP (3) regardless of the network load and the traffic ratio. This is not the case in HCBR, as it fails to perform better than Dijkstra (3)-WSP (3) with traffic ratio 3:1 under load < 0.6 and with traffic ratio 2:1 under load < 0.4. The path selection method used in HBBR performs well under varying network loads when compared to HCBR and the Dijkstra-WSP algorithms with small update intervals in heterogeneous networks.

(a) Ratio 3:1

(a) Ratio 2:1

Figure 7.4 Flow blocking probability in heterogeneous network

7.3.6 Impact of network topologies

In figure 7.5 the flow blocking probability is plotted against different ranges of arrival rate for different types of network topologies. We start by offering a small load, 0.5, whereby all algorithms can accept most of the connections since with very small offered load it is easy to find a path that satisfies the requested bandwidth. We then increase the load to see how the local and global algorithms will perform. From this figure we can notice that in RAND400 with ratio 4:1 and 3:1, HCBR and HBBR schemes give superb performance compared with Dijkstra-WSP algorithms, even with a small interval update (3) of global state information. However, because of path selection in the HBBR algorithm, which relies on the residual bandwidth on a path, its blocking probability is better than HCBR again in all topologies. This can be noticed with ratio 3:1 when HBBR performs almost as Dijkstra-WSP with zero update intervals. As the load increases, HCBR and HBBR adapt to the change and maintain their relative performance. This can be seen in RAND400 and heterogeneous topologies, as the blocking probability increases gradually, whereas the Dijkstra-Wsp algorithm can't react promptly to changes in load and perform poorly as the load increases. This can be expected as the periodic updates do not respond quickly enough to rapid variations in load. In the case of the lattice topology, HBBR and HCBR fail to perform better than Dijkstra-WSP (3) although HBBR gives good performance with low loads, but still gives good results against Dijkstra-WSP (15). This is most likely because the

Lattice is a regular topology and there would thus be less likelihood of route flapping with Dijkstra-Wsp algorithms.

(a) RAND 400 – Ratio 4:1

(b) RAND 400 – Ratio 3:1
Figure 7.5 Impact of network topology

(c) Lattice – Ratio 1:1

Figure 7.5 Impact of network topology (Continued)

7.3.7 Impact of varying non-uniform traffic

So far the destination nodes have been chosen from a uniform random distribution. However, on the internet the areas are partitioned into smaller units because of the resource requirements for the network's management and also to decrease the path cost of computation. Moreover, some source nodes may receive more flows to specific destination nodes. It is also emphasized that the uniform end-to-end IP QoS solution is not realistic [97]. For these reasons, we have partitioned these areas, so flows routed in the same blocks of a partition are three times more frequent than flows routed across blocks. We have divided each area in HBBR and HCBR based on the average length of the candidate path set between the border node in the area and the other nodes in the same area.

Similarly, in Dijkstra the nodes with the least distance to the border node were grouped in one block and the other in the second block.

(a) Non-uniform traffic in low level – ratio 3:1

(b) Non-uniform traffic in the backbone– ratio 2:1

Figure 7.6 Impact of varying non-uniform traffic in heterogeneous topology

In figure 7.6(a) the flow blocking probability is plotted against different offered loads in the heterogeneous topology with ratio 3:1 using uniform and non-uniform traffic in these areas. It can be noticed that under non-uniform traffic all algorithms perform well, compared to uniform traffic. This is due to the fact that a source node needs to have more up-to-date information for frequent destinations; in the case of Dijkstra's algorithm larger hop counts may be needed to find the widest path. Therefore, smaller areas decrease the overhead of finding a feasible path. In contrast, larger areas need more QoS updates to be exchanged, which increases the overhead in the network. However, since a source node in the localized QoS routing mechanism collects statistics about the network state, it does have more accurate information about frequently used destination nodes. These results illustrate the ability of localized QoS routing algorithms to perform better than global QoS routing algorithms in terms of non-uniform traffic without increasing overhead in the network.

On the internet it is likely some subnetworks receive more traffic and some may receive less traffic than the other subnetworks. Hence source nodes need more accurate information about the QoS state to reach these subnetworks. In case of global QoS routing more frequent updates of the QoS state are required and this would increase overhead on the network. In the heterogeneous topology the loads offered to random 60 and random 80 are three times more than the loads offered to the other subnetworks. The top level in the topology will be overloaded by

more traffic to random 60 and random 80, so we can see how the localized and global algorithms will react with the extra load. As we can see from figure 7.6(b), the performance of HBBR, HCBR and WSP degrade significantly, but this time the uniform traffic loading gives better results. This is expected since most of the traffic in the non-uniform case is routed to specific subnetworks resulting in an imbalanced loading in the backbone. It is therefore hard to find a path to satisfy the QoS bandwidth through these subnetworks; however, we can notice that HBBR performs well under low loads even with extra load on the backbone, which implies the effectiveness of path selection that explicitly reflects the quality of a path.

7.3.8 Impact of bursty traffic

We model bursty traffic in the network to study the effect of bursty connection arrivals on localized and global routing algorithms. Following [1] [96], the burstiness of traffic is modelled using a Weibull distribution with two different values of shape parameter of the distribution, 0.2 and 0.7, where burstiness is increased with a smaller shape value. Figure 7.7 shows the blocking probability plotted against offered loads, from 0.4 to 0.8, with two shape values for each algorithm in heterogeneous and lattice topologies. As can be noticed, the blocking probability for the Dijkstra-WSP algorithm in figure 7.7 (bursty) in both topologies with update intervals of 3 is significantly higher than that in figure 7.4

and figure 7.5 (Poisson) for both shapes, particularly in the lattice topology. This can be expected because bursty connection arrivals increase blocking by making it harder for the source node to find feasible paths compared to Poisson traffic, even for small update intervals.

In the hierarchical localized QoS routing algorithm (HBBR), the blocking probability for bursty traffic also increases compared to non-bursty traffic, but not as high as Dijkstra-WSP(3) routing algorithms. This is because a source node in a localized routing algorithm makes its routing decision based on its view of the network, not on the global state information. On the other hand, HBBR gives superior performance under bursty traffic with shape 0.7; it has the least blocking probability in both topologies. Good performance is also found with shape 0.2, which is burstier, with less blocking probability than Dijkstra-WSP (3) with shape 0.7, which is less bursty and HCBR with shape 0.7 under low load. This is obvious, since HBBR takes routing decisions based on average residual bandwidth in the network areas and the network backbone. The path selection process in HBBR, which is based directly on the required QoS metric (residual bandwidth) reflects on the quality of the path. It can also be noticed that the difference in Dijkstra (3)-WSP (3) in the two shapes is large compared to the difference in localized algorithms, which suggests that localized routing schemes are not significantly affected by bursty traffic in large networks.

(a) Lattice topology: ratio 1:1

(b) Heterogeneous ratio 3:1

Figure 7.7 Impact of bursty traffic

7.3.9 Impact of heterogeneous traffic

The experiments so far have seen only small bandwidth [0.1-2] being used as the QoS bandwidth. In the following we study the effect of using small and large bandwidth on Dijkstra-WSP, HCBR and HBBR in the lattice topology. We consider two ranges of bandwidth in order to study the impact of large and small bandwidth flows, but having the same holding time. The amount of bandwidth requested for both types are uniformly chosen from the range 2 to 4 for the large flows and 0.1 to 2 for the small flows, with the mean 3 and 1.05 respectively. The holding times for all flows are exponentially distributed with the mean 56.223. The performance is measured by mixing fractions of small and large flows.

Figure 7.8 Impact of heterogeneous traffic in lattice topology

Figure 7.8 shows flow blocking probability plotted against the fraction of small bandwidth flows. It can be noticed that the blocking probability of all hierarchical algorithms is decreased as the fraction of small bandwidth flows increases, which is expected since it is easier for a source node to find a feasible path with a small QoS bandwidth. Dijkstra (3)-WSP (3) gives poor performance, unless most of the load is small bandwidth, which is not the case in HBBR which gives a better performance. This can be expected, as HBBR continuously monitors the candidate path residual bandwidth, and the requested QoS bandwidth is known before path selection for both large and small bandwidths. It can be noticed that Dijkstra (3)-WSP (3) gives poor performance with large bandwidths, as Dijkstra in the low level seeks the widest path and the algorithm consumes more recourses searching for large bandwidths. HCBR however lies in the middle as it gives good performance when most of the fraction is small. It's blocking probability increases as the fraction of large bandwidth increases but the increase is less than with the Dijkstra algorithm.

7.3.10 HBBR Stability

Unlike global routing algorithms which exhibit route flapping due to exchanging global information [98] [1], localized routing algorithms do not exhibit such behaviour, as can be noted from figure 7.9. Localized routing algorithms are stable as routing decisions are taken based on each source node’s view of the

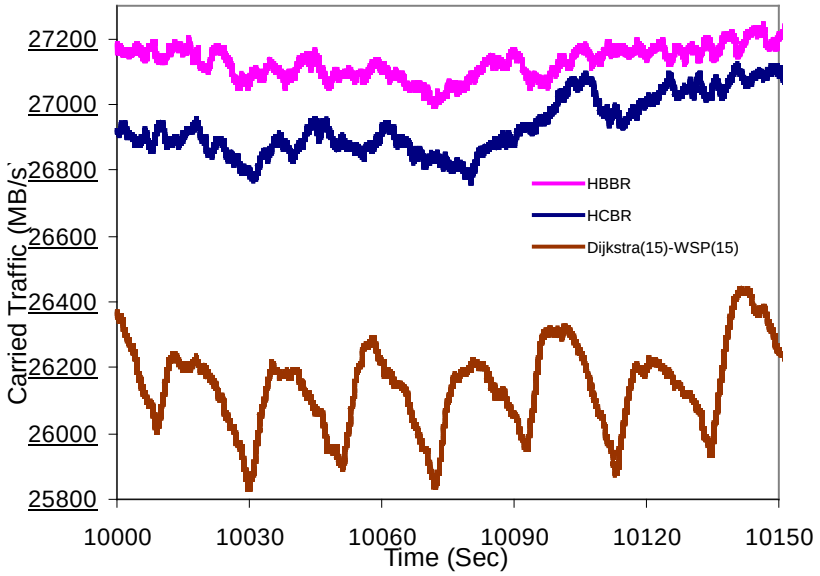


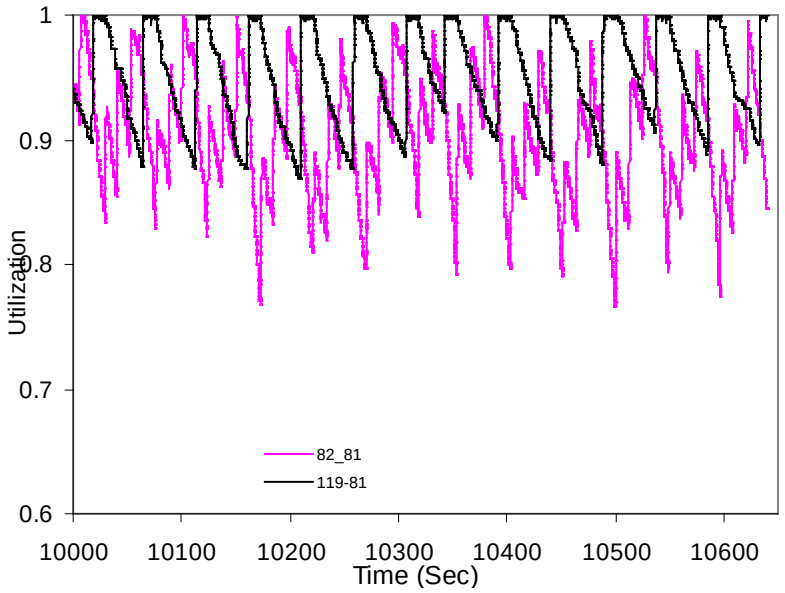
Figure 7.9 Carried load fluctuation

network. We can also observe that HBBR can carry more bandwidth than HCBR and Dijkstra (15)-WSP (15) because its blocking probability is lower.

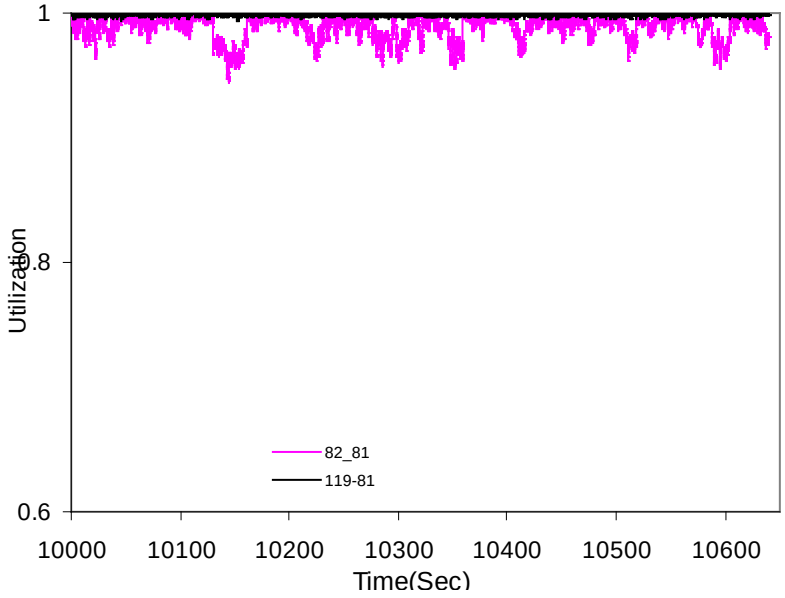
A good QoS routing method should efficiently utilize network resources and have a small overhead in exchanging QoS state information. In global QoS routing large update intervals of QoS information may lead to route flapping [70]. Such behaviour occurs when the utilization on a link is low and the out-of-date information causes all nodes to route traffic along this link, resulting in rapid utilization of this link. Likewise, with high utilization, all source nodes avoid using this link and its utilization decreases [74]. Such oscillatory behaviour results in poor route selection, instability and an overall degradation of network performance.

We observed the links that connect random 40 with the backbone in the heterogeneous topology (figure 7.2) over a period of time under Dijkstra (30), HCBR and HBBR to illustrate this behaviour. The traffic in the topology is directed from source nodes in the random 40 subnetwork to the other subnetwork nodes. The capacities of the links $82 \rightarrow 81$ and $119 \rightarrow 81$ is set to 150 and the mean holding time for flows is set to 100 sec. Figure 7.10(a) shows the oscillation in the Dijkstra algorithm as when the utilization on the link $119 \rightarrow 81$ goes high, the utilization of the link $82 \rightarrow 81$ goes low. This is because all source nodes in random 40 route traffic to the same link until the next update interval indicates the overload of that link. After the source nodes have the new state they flap routes to the other link.

However, the fluctuation in link utilization with HCBR and HBBR in figure 7.10(b) and figure 7.10(c) is much smaller than Dijkstra. This is because localized QoS routing algorithms select a path based on their local view of the network state. It is also worth noting that the fluctuation in HBBR is less than that in HCBR which suggests the stability of HBBR is likely due to the selection of the path based on its quality, so it distributes the load between paths based on the residual bandwidth.

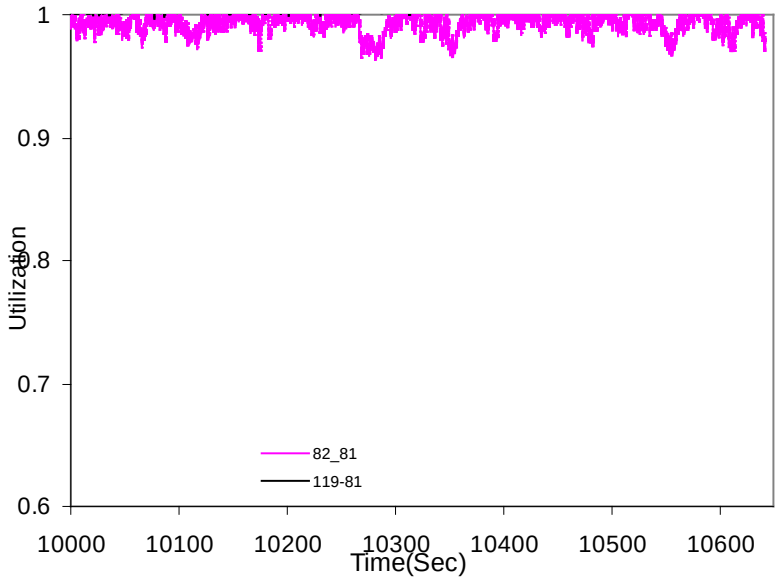


(a) Dijkstra



(b) HCBR

Figure 7.10 Fluctuation in utilization of two links in heterogeneous topology



(c) HBBR

Figure 7.10 Fluctuation in utilization of two links in heterogeneous topology

(Continued)

It is important for good routing algorithms to be able to stabilize and adapt to rapid change in the network state, since out-of-date information in global routing algorithms that is collected about the network state may cause route flapping with a large update interval. The blocking probability under Dijkstra (15)-WSP (15), HCBR and HBBR as a function of time is plotted in figure 7.11. We consider three levels of offered load to the lattice topology; where a load of 0.7 is offered to the network and then increased to 0.8 and decreased to 0.6. It can be seen that under all these arrivals, HCBR and HBBR adapt quickly to a rapid change in arrivals and their blocking probability stabilizes after a short time of fluctuations

Figure 7.11 Response to rapid arrival variation in lattice topology

when compared to Dijkstra (15)-WSP (15). This is due to the fact that each source node performs routing based on its local view of the network state. So we can claim that unlike global routing algorithms, which exhibit route flapping due to exchanging global information, localized QoS routing algorithms are more stable than global QoS routing algorithms and do not exhibit such behaviour.

7.3.11 Load Balancing

Figure 7.12 compares the average Jain's index of HBBR, HCBR and Dijkstra (15)-WSP (15) in the heterogeneous topology. The index is calculated for the low level figure 7.12(a) and the backbone figure 7.12(b) with the ratio 3:1 and 1:1 respectively. The values for the index reflect an average taken at regular intervals

of time over 2,000,000 connection requests since the load fluctuates over time. The average Jain's index is plotted as a function of offered load. From the figures, we see that HCBR gives the least fairness in both levels, which is expected since HCBR always tries to select the path with highest credits regardless of the bandwidth in the path. As the load increases, we can notice that the fairness is increased, which is also expected since any flow rejection decrease will decrease the path credit and an alternative path will be selected.

Unlike HCBR, which sticks with the same path as long as this path has the most credits, HBBR distributes the load among the candidate paths. This can be noticed as it gives the best fairness index in both levels. HBBR selects the path based on its residual bandwidth and may select the alternative without flow rejection. As a result of its path selection mechanism, it gives a good fairness index regardless of the load in the network. On the other hand, Dijkstra behaves like HCBR in the low levels as its fairness index increases as the load increases and this is because Dijkstra searches for the widest path. WSP however gives good performance in the top level and is not greatly affected by load change, which is expected since WSP always selects the shortest path and if there is more than one it selects the widest one. Hence it distributes the load among these paths as their carried loads change.

(a) Low level – Ratio3: l

(b) Top level – Ratio1: l

Figure 7.12 Average Jain's index in heterogeneous topology

7.3.12 HBBR sensitivity to W parameter

Since HBBR monitors the residual bandwidth in each path of the candidate path sets, a sliding window W with a predetermined period is used to record residual bandwidth upon flow acceptance or rejection. In figure 7.13, flow blocking probability is plotted against different values of window size, with the fixed load 0.6 in the heterogeneous topology. It was found that the blocking probability decreases slightly as the value of W increases. This parameter controls the observation period of the path residual bandwidth; so a longer period is better to get a good estimation of how good or bad the path is.

Figure 7.13 Window size (connection requests) in heterogeneous topology

7.3.13 *HBBR time complexity*

Global QoS routing algorithms performing a variant of Dijkstra's algorithm to find the shortest path or widest path that we used in the simulation, take at least O (N log $N+L$) time; where N is the number of nodes and L is the number of links in the network. On the other hand, the complexity of selecting a path among the set of candidate paths R in HBBR and HCBR is O (|R|). HCBR need to update blocking probabilities, which takes a constant time O (1). HBBR also needs to update the average residual bandwidth, which also takes O (1).

7.4 Conclusions

We have developed a hierarchical bandwidth based localized QoS routing algorithm HBBR, which relies on actual residual bandwidth on the path in order to take routing decisions. We demonstrated through simulation that the proposed algorithm, although simple, performs better than HCBR under different traffic loads in a two level hierarchical topologies and outperforms global routing schemes except when global schemes have unrealistically small update intervals. Our results suggest that:

- Hierarchical localized routing schemes, which explicitly reflect the quality of a path, are more suitable than the global routing schemes in hierarchical networks and therefore might usefully be employed on the internet.
- Hierarchical localized QoS routing can be employed to advantage in a network with different requirements or heterogeneous traffic like the internet.
- Hierarchical localized QoS routing performs well with non-uniform traffic patterns and would lend itself to a partitioned network where the partition reflects the non-uniformity.
- Hierarchical localized QoS routing also performs better than global schemes with bursty connection requests which again suggests it would be suited to the type of traffic patterns found on the internet.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

In QoS routing, knowledge about the global network state is critical in path selection. However, with the rapid changes in network resources required with each new connection request, maintaining an accurate network QoS state is not practical. This is due to the unreasonable communication and processing overheads caused by frequent exchange of QoS state information. However, reducing the exchange rate leads to inaccurate information of the global network QoS state due to stale resource information and this degrades the performance of QoS routing schemes.

As an alternative to the global QoS routing approach, a localized QoS routing approach has been proposed to solve the inherent scalability problem. In such an approach, a source node infers the network state from flow statistics collected locally. Such an efficient approach for QoS routing has been the topic of this thesis and we contribute a number of new insights into localized QoS routing as follows.

- Throughout this thesis, we have proposed four localized QoS routing algorithms. Each of them is compared with global QoS routing algorithms and it is shown that localised QoS routing can be considered a viable alternative scalable approach to the use of global QoS routing algorithms.
- It has been demonstrated throughout the thesis that localized QoS routing should be based on schemes that explicitly reflect the quality of a path, rather than schemes that are based indirectly on the path quality.
- It has also been demonstrated throughout the thesis that localized QoS routing performs well with non-uniform traffic patterns and would lead itself to partitioned networks that reflect the non-uniformity.
- It has also been demonstrated throughout the thesis that localized QoS routing performs better than global schemes with bursty connection requests, different requirements or heterogeneous traffic; therefore it might usefully be employed on the internet.

8.2 Future Works

Possible directions for future work include the following:

- Although the algorithms proposed in this thesis use bandwidth (BBR) or delay (DBR & CBR) for the flat schemes as the only QoS metric, it appears that combining delay and bandwidth would be good for future related work.
- In hierarchical localized QoS routing (HBBR) we considered bandwidth as the only QoS metric. This can be modified to other QoS metrics such as delay or cost. Another feasible extension to the HBBR is combining the bandwidth metric with the other QoS metrics.
- Another possible extension to localized QoS routing would be to use it in the case of multicast QoS routing which has not been studied in the literature before.
- Since the performance of localized QoS routing algorithms depends on the selection of candidate path set, this needs more investigation in the context of both bandwidth and delay. Also, selection of a candidate path set to suit a specific QoS metric might be looked into.
- Localized QoS routing maintain a constant set of candidate path between any source and destination node in the network. However in the real network a link failure may occur and this would have a potential impact on the network performance. So it is important to investigate maintaining a

dynamic set of candidate paths between any source and destination in the network.

- Another feasible extension to localized QoS routing would be to use it as a Load balancing algorithm for optimizing the usage of network resources. The current implementation of localized QoS routing always attempts to send the QoS connection request to the most feasible candidate path among the candidate path set. Localized QoS routing approaches might be enhanced by having the load balancing implemented between all the candidate paths of the same candidate path set. The load balancing can be in the context of available bandwidth on a candidate path, or end-to-end delay on a candidate path.

References

[1] A. Shaikh, J. Rexford, and K. Shin, "Evaluating the Impact of Stale Link State on Quality-of-Service Routing," IEEE/ACM Transactions on Networking, vol. 9, pp. 162--176, 2001.

[2] S. Alabbad and M. E. Woodward, "Localized Credit Based QoS Routing: Performance Evaluation Using Simulation," in the 40th Annual Simulation Symposium, Huntsville, AL 2006.

[3] A. S. Tanenbaum, Computer networks, 4th ed. Upper Saddle River, N.J.: Prentice Hall PTR, 2003.

[4] D. P. Bertsekas and R. G. Gallager, Data networks, 2nd ed. Englewood Cliffs, N.J.: Prentice Hall, 1992.

[5] J. Moy, "OSPF Version 2," in Internet Request for Comments (RFC 2328), 1998.

[6] G. Malkin, "RIP version 2," in IETF RFC 2453 1998.

[7] J. Postel, "Transmission control protocol," in IETF RFC 793, 1981.

[8] J. Postel, "User datagram protocol," in IETF RFC 768, 1980.

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

[10] X. Masip-Bruin. et al, "Research Challenges in QoS Routing," Computer Communications, vol. 29, pp. 563-581, 2006.

[11] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: An Overview," in IETF RFC 1633, 1994.

[12] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation Protocol (RVSP) – Version 1 Functional Specification," in IETF RFC 2205, 1997.

[13] S. Blake. et al, "An Architecture for Differentiated Services," in IETF RFC 2475, 1998.

[14] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," in Internet Draft<draft-ietf-mpls-arch-01.txt>, 1998.

[15] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McMaus, "Requirements for Traffic Engineering over MPLS," in Internet Draft, <draftietf-mpls-traffic-eng.00.txt>, 1998.

[16] D. Durham and R. Yavatkar, "Inside the Internet's Resource ReSerVation Protocol: Foundations for Quality of Service," in John Wiley & Sons, 1999.

[17] J. Heinanen, F. Baker, W.Weiss, and J. Wroclawski, "Assured Forwarding PHB Group," in RFC 2297, 1999.

[18] V. Jacobson, K. Nichols, and K. Poduri, "An Expedited Forwarding PHB," in IETF RFC 2298, 1999.

[19] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, "Requirements for Traffic Engineering over MPLS," in RFC 2702, 1999.

[20] Z. Wang, Internet QoS: Architectures and Mechanisms for Quality of Service: Morgan Kaufmann, 2001.

[21] P. Zhang and R. Kantola, "Building MPLS VPNs with QoS Routing Capabilitypp," in Fifth International Symposium on Interworking, pp. 292–301, 2000.

[22] O. Younis and S. Fahmy, "Constraint-Based Routing in the Internet: Basic Principles and Recent Research," IEEE Communications Surveys & Tutorials, vol. 5, pp. 2-13, 2003.

[23] S. Chen and K. Nahrstedt, "An Overview of Quality-of-Service Routing for the Next Generation HighSpeed Networks: Problems and Solutions," IEEE Network Magazine, vol. 12, pp. 64 -79, 1998.

[24] H. Lu and I. Faynberg, "An Architectural Framework for Support of Quality of Service in Packet Networks," IEEE Communications Magazine, pp. 98-105, 2003.

[25] Z. Whang and J. Crowcroft, "Quality-of-Service routing for supporting multimedia applications," IEEE J. Select. Areas Communication, vol. 14, pp. 1228-1234, 1996.

[26] S. Nelakuditi, R. Tsang, and Z. Zhang, "Quality-of-Service Routing without Global Information Exchange," in IWQOS, 1999.

[27] A. Orda and A. Sprintson, "Precomputation Schemes for QoS Routing," IEEE/ACM Transactions on Networking, vol. 11, pp. 578-591, 2003.

[28] G. Apostolopoulos and S. K. ripathi, "On reducing the processing cost of on-demand QoS path computation," Journal of High Speed Networks, vol. 7, pp. 77-98, 1998.

[29] D. Medhi, "QoS Routing Computation with Path Caching: A Framework and Network Performance," IEEE Communications Magazine, vol. 40, pp. 106-113, December 2002.

[30] H.Zhu, "Comparison between Pre-Computation and On-Demand Computation QoS Routing with Different Link State Update Algorithms." vol. Master's thesis: Helsinki University of Technology, 2003.

[31] N. Ansari, G. Cheng, and N. Wang, "Routing-oriented update scheme (ROSE) for link state updating," IEEE Transactions on Communications, vol. 56, pp. 948-956, 2008.

[32] B. Fu, F. A. Kuipers, and P. V. Mieghem, "To update network state or not? ," in the 4th international telecommunication networking workshop on QoS in multiservice IP networks, Feb 2008.

[33] A. Ariza, E. Casilari, and F. Sandoval, "Strategies for updating link states in QoS routers," Electronic Letters, vol. 36, 2000.

[34] B. Lekovic and P. V. Mieghem, "Link state update policies for Quality of service routing," in Eighth IEEE Symposium on Communications and Vehicular Technology in the Benelux (SCVT2001), 2001.

[35] M. R. Garey and D. S. Johnson, Computers and intractability : a guide to the theory of NP-completeness. San Francisco: W. H. Freeman,, 1979.

[36] A. Striegel and G. Manimaran, "A Survey of QoS Multicasting Issues," IEEE Communications Magazine, vol. 40 i6, 2002.

[37] F. A. Kuipers and P. F. A. V. Mieghem, "Conditions That Impact the Complexity of QoS Routing," IEEE/ACM Transactions on Networking, vol. 13, pp. 717-730, 2005.

[38] Q. Ma and P. Steenkiste, "Quality-of-Service Routing for Traffic with Performance Guarantees," in Proceedings of the IFIP Fifth International Workshop on Quality of Service, New York, pp. 115-126, 1997.

[39] G. Chen and Y. Tian, "A New QoS Routing Framework for Solving MCP," IEEE Transaction on Communications, 2003.

[40] X. Yuan, "Heuristic Algorithms for Multi-Constrained Quality of Service Routing," in IEEE/ACM Transactions on Networking, pp. 244-256, 2002.

[41] X. Yuan, "On the extended Bellman-Ford algorithm to solve two-constrained quality of service routing problems," in Proceedings of the Eighth International Conference on Computer Communications and Networks, pp. 304-310, 1999.

[42] S. Chen and K. Nahrsted, "On finding multi-constrained path," in IEEE ICC'98, pp. 874-899, 1998.

[43] C. Hendrick, "Routing Information Protocol," in IETF RFC 1058, 1998.

[44] G. Apostolopoulos. et al, "Intradomain QoS Routing in IP Networks: A Feasibility and Cost/Benefit Analysis," IEEE Network (Special Issue on Integrated and Differentiated Services for the Internet), vol. 13, pp. 42-54, 1999.

[45] G. Apostolopoulos, D. Williams, S. Kamat, R. Guerin, A. Orda, and T. Przygienda., "QoS Routing Mechanism and OSPF Extensions," in RFC 2676, 1999.

[46] Y. Yang, L. Zhang, J. K. Muppala, and S. T. Chanson, "Bandwidth Delay Constrained Routing Algorithms," Computer Networks, vol. 42, pp. 503-520, 2003.

[47] S. S. Deb and M. E. Woodward, "A New Approach to Scale QoS Routing Algorithms," in Proc. of IEEE GLOBECOM'04,, Dallas, USA, 2004.

[48] D. H. Lorenz and A. Orda, "Optimal Partition of QoS Requirements on Unicast Paths and Multicast Trees," IEEE/ACM Transactions on Networking, vol. 10, pp. 102-114, 2002.

[49] S. Srivastava. et al, "Benefits of Traffic Engineering using QoS Routing Schemes and Network Controls," Computer Communications Journal, vol. 27, pp. 387-399, 2004.

[50] J. L. Sobrinho, "Algebra and Algorithms for QoS Path Computation and Hop-by-Hop Routing in the Internet," IEEE/ACM Transactions on Networking, vol. 10, pp. 541-550, 2002.

[51] D. S. Reeves and H. F. Salama, " A Distributed Algorithm for Delay-Constrained Unicast Routing," IEEE/ACM Transactions on Networking, April 2000.

[52] S. Chen and K. Nahrstedt, "Distributed Quality-of-Service routing in HighSpeed Networks Based on Selective Probing," in 23rd Annual Conference on Local Area Networks (LCN'98, pp. 80--89), 1998.

[53] D. Ghosh, V. Sarangan, and R. Acharya, "Quality-of- Service Routing in IP networks," IEEE Transactions on Multimedia, vol. 3, June 2001.

[54] B. Awerbuch, B. K. Y. Du, and Y. Shavitt, "Routing Through Teranode Networks with Topology Aggregation," in IEEE ISCC'98, Athens, Greece, 1998.

[55] J. Bhhrens and J. J. Garcia-Luna-Aceves, "Hierarchical Routing Using Link Vectors," in IEEE Infocom'98, 1998.

[56] R. Izmailov, A. Iwata, B. Sengupta, and H. Suzuki, "PNNI Routing Algorithms for Multimedia ATM Internet," NEC Research and Development, vol. 38, pp. 60-73, 1997.

[57] A. Iwata and N. Fujita, "A Hierarchical Multilayer QoS Routing System with Dynamic SLA Management," IEEE Journal on Selected Areas in Communication, vol. 18, pp. 2603–2616, December 2000.

[58] T. Korkmaz and M. Krunz, "Source-oriented topology aggregation with multiple QoS parameters in hierarchical networks," ACM Transactions on Modeling and Computer Simulation, vol. 10, pp. 295--325, 2000.

[59] P. Paul and S. V. Raghavan, "Survey of qos routing," in 15th international conference on Computer communication, Mumbai, India, pp. 50-75, 2002.

[60] M. Curado and E. Monteiro, "A Survey of QoS Routing Algorithms," in the International Conference on Information Technology (ICIT2004), Istambul, Turkey, December 2004.

[61] A. Gibbons, Algorithmic Graph Theory. Cambridge: Cambridge University Press, 1985.

[62] E. W. Dijkstra, "A note on two problems in connexion with graphs," Numerische Mathematik, pp. 1: 269-271, 1959.

[63] R. Bellman, "On a routing problem," Quarterly of Applied Mathematics, vol. 16(1), pp. 87-90, 1958.

[64] Q. Ma and P. Steenkiste, "On path selection for traffic with bandwidth guarantees," in 5th IEEE International Conference on Network Protocols, Atlanta, GA, pp. 191--202, 1997.

[65] E. I. Chong, S. Maddila, and S. Morley, "On finding single-source single-destination k shortest paths," Journal of Computing and Information, pp. 40-47, 1995.

[66] B. Peng, A. H. Kemp, and S. Boussakta, "QoS Routing with bandwidth and Hop-Count Consideration: A Performance Perspective," Journal of Communications, vol. 1, 2006.

[67] A. A. Shaikh, "Efficient Dynamic Routing In Wide-Area Networks
" in Dept. of Computer Science and Engineering Ann Arbor: The University of Michigan, 1999.

[68] I. Matta and A. U. Shankar, "Dynamic routing of real-time virtual circuits," in Proceedings of IEEE International Conference on Network Protocols, Columbus, OH, pp. 132–139, 1996.

[69] A. Shaikh, J. Rexford, and K. Shin, "Evaluating the Overheads of Source-Directed Quality of Service Routing," in Proceedings of the International Conference on Network Protocols (ICNP), 1998.

[70] A. Shaikh, J. Rexford, and K. Shin, "Efficient Precomputation of Quality of Service Routes," in International Workshop on Network and OS Support for Digital Audio and Video (NOSSDAV '98) Cambridge, 1998.

[71] Q. Ma, P. Steenkiste, and H. Zhang, "Routing High-bandwidth Traffic in Max-min Fair Share Networks," in Proceedings of ACM SIGCOMM'96, Palo Alto, CA, pp. 206--217, 1996.

[72] J. Wang and K. Nahrstedt, "Hop-by-Hop Routing Algorithms For Premium-class Traffic In DiffServ Networks," in Infocom. vol. 2, 2002.

[73] S. Nelakuditi, "Localized approch to providing quality-of-service," in Dept. of Computer Science and Engineering Minneapolis: University of Minnesota, p. 174, 2001.

[74] S. Nelakuditi, Z. L. Zhang, R. Tsang, and D. Du, "Adaptive Proportional Routing: a Localized QoS Routing Approach," IEEE/ACM Transactions on Networking, vol. 10, pp. 790--804, 2002.

[75] X. Yuan and A. Saifee, "Path Selection Methods for Localized Quality of Service Routing," in The 10th IEEE International Conference on Computer Communications and Networks (IC3N 2001), Phoenix, Arizona, pp. 102-107, 2001.

[76] S. Nelakuditi, Z.-L. Zhang, R. Tsang, and D. Du, "On selection of candidate paths for proportional routing," Computer Networks, vol. 44, pp. 79-102 2004.

[77] R. J. Gibbens, F. P. Kelly, and P. B. Key, "Dynamic Alternative Routing: Modelling and Behaviour," in 12th Int. Teletraffic Congress, Torino, pp. 3.4A.3.1--3.4A.3.7, 1988.

[78] D. Mitra and J. B. Seery, "Comparative Evaluations of Randomized and Dynamic Routing Strategies for Circuit-Switched Networks," IEEE Trans. on Communications, vol. 39, 102-116 1991.

[79] K. S. Narendra and P. Mars, "The Use of Learning Algorithms in Telephone Traffic Routing - A Methodology," Automatica, vol. 19, pp. 495-502, 1983.

[80] S. Alabbad and M. E. Woodward, "Localized Credit Based QoS Routing," Proceedings of IEE, vol. 153, pp. 787-796, 2006.

[81] V. Paxson and S. Floyd, "Why we don’t know how to simulate the internet," in Proceedings of the 1997 Winter Simulation Conference, 1997.

[82] R. Pastos-Satorras and A. Vespignani, Evolution and structure of the Internet. Cambridge: University Press, Cambridge, 2004.

[83] P. Erdos and A. renyi, "On Random Graph " Publ. Math, pp. 290-297, 1959.

[84] B.M.Waxman, "Routing of multipoint connections," IEEE J. Select. Areas Communications, vol. 6(9), pp. 1617-1622, 1988.

[85] M. Doar and I. Leslie, "How Bad is Naïve Multicast Routing?," IEEE INFOCOM pp. 82-89, 1993.

[86] W. C. Lee, "Topology aggregation for hierarchical routing in ATM networks," ACM SIGCOMM Compuer Communication Review, vol. 25, pp. 82-92, 1995.

[87] G. Apostolopoulos, R. Guerin, and S. Kamat, "Implementation and performance measurements of QoS routing extensions to OSPF," in Proceedings of IEEE INFOCOM, New York, 1999.

[88] A. Varga, "OMNeT++ Community Site," in URL:http://www.omnetpp.org, 2009.

[89] A. Varga, "OMNeT++ Discrete Event Simulation System Version 3.2 User Manual," in URL:http://www.omnetpp.org/doc/manual/usman.html, 2008.

[90] A. Varga, "The OMNeT++ Discrete Event Simulation System," in the European Simulation Multiconference, Prague, Czech Republic, 2001.

[91] R. Jain, D. Chiu, and W. Hawe, "A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems," DEC Research Report TR-301 1984.

[92] A. M. Law and W. D. Kelton, Simulation Modeling and Analysis, 3rd ed. New York: McGraw-Hill, 2000.

[93] B. Peng, A. Kemp, and S. Boussakta, "Impact of network conditions on QoS routing algorithms," in 3rd IEEE Consumer Communications and Networking, pp. 25-29, 2006.

[94] H. Pung, J. Song, and L. Jacob, "Fast and efficient flooding based QoS routing algorithm," in IEEE ICCCN99, pp. 298--303, 1999.

[95] A. Feldmann, "Impact of non-poisson arrival sequences for call admission algorithms with and without delay," in GLOBECOM'96. IEEE Global Telecommunications Conference, London, UK, pp. pp. 617--622, 1996.

[96] A. Feldmann, "Characteristics of TCP connections arrivals," in Self-similar Network Traffic and Performance Evaluation, K. Park and W. Willinger, Eds. New York: John Wiley and Sons, pp. 367--399, 2000.

[97] J. Soldatos, E. Vayias, and G. Kormentzas, "On the building blocks of quality of service in heterogeneous IP networks," IEEE Communications Surveys and Tutorials, vol. 7, 2007.

[98] A. Shaikh, J. Rexford, and K. Shin, "Load-Sensitive Routing of Long-Lived IP Flows," in ACM SIGCOMM, 1998.

[99] S. Alabbad and M. E. Woodward, "Localized Route Selection for Traffic with Bandwidth Guarantees," SIMULATION, vol. 83, pp. 259-272, 2007.

[100] A. Forum, "Private Network-Network Interface(PNNI) v1.0 specifications," 1996.

[101] Y. Tang and S. Chen, "QoS information approximation for aggregated networks," in IEEE International Conference on Communications, 2004.

[102] K.-S.Lui, K. Nahrstedt, and S. Chen, "Routing with Topology Aggregation in Delay-Bandwidth Sensitive Networks," IEEE Transactions on Networking, vol. 12, pp. 17-29, 2004.

[103] S. Zarifzadeh, A. Nayyeri, N. Yazdani, and C. Lucas, "ADAM: An Adaptive Model for State Aggregation in Hierarchical Networks," in Asia-Pacific Conference on Communications, 2005.

[104] Y. Wang and Z. Wang. Explicit Routing Algorithms for Internet Traffic Engineering. In Proceedings of ICCCN'99, Sept. 1999.

[105] R. Callon. Use of OSI IS-IS for routing in TCP/IP and dual environments. IETF RFC 1195, December 1990.

[106] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). RFC 1771, March 1995.

[107] Nelakuditi, S., Zhang, Z.: 'Localized Quality of Service Routing for the Internet'(Kluwer Academic Publishers, 2003).

[108] E. Gelenbe, E. Seref, Z. Xu " Towards networks with intelligent packets, "Proceedings of the Fourteenth International Symposium on Computer and Information Sciences, pp. 1-11, Oct. 1999.

[109] E. Gelenbe, R. Lent, A. Montuori, Z. Xu "Cognitive packet networks: QoS and performance", Opening Keynote Paper, IEEE MASCOTS'02 Conference (IEEE Computer Society), Ft. Worth, TX, pp. 3-12, Oct. 2002

[110] S. Chen and K. Nahrstedt, "Distributed QoS Routing with Imprecise State Information," in 7th IEEE International Conference on Computer, Communications and Networks, Lafayette, LA, pp. 614-621, 1998.

[111] W. Usaha, J. Barria, A reinforcement learning ticket-based probing path discovery scheme for MANETs, Ad Hoc Networks, Vol: 2, Pages: 319 - 334, 2004.