

# bradscholars

**Novel localised quality of service routing algorithms.  
Performance evaluation of some new localised quality  
of service routing algorithms based on bandwidth  
and delay as the metrics for candidate path selection.**

Item Type	Thesis
Authors	Alghamdi, Turki A.
Rights	<a href="http://creativecommons.org/licenses/by-nc-nd/3.0/">&lt;a rel="license" href="http://creativecommons.org/licenses/by-nc-nd/3.0/"&gt;&lt;img alt="Creative Commons License" style="border-width:0" src="http://i.creativecommons.org/l/by-nc-nd/3.0/88x31.png" /&gt;&lt;/a&gt;&lt;br /&gt;</a> The University of Bradford theses are licenced under a <a href="http://creativecommons.org/licenses/by-nc-nd/3.0/">&lt;a rel="license" href="http://creativecommons.org/licenses/by-nc-nd/3.0/"&gt;Creative Commons Licence&lt;/a&gt;.</a>
Download date	2026-03-06 00:15:52
Link to Item	<a href="http://hdl.handle.net/10454/5420">http://hdl.handle.net/10454/5420</a>



## **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](#).

# **NOVEL LOCALISED QUALITY OF SERVICE ROUTING ALGORITHMS**

**Turki A. ALGHAMDI**

PhD

Department of Computing  
School of Computing, Informatics and Media  
University of Bradford

2010

# **NOVEL LOCALISED QUALITY OF SERVICE ROUTING ALGORITHMS**

Performance Evaluation of Some New Localised Quality of  
Service Routing Algorithms Based on Bandwidth and Delay as  
the Metrics for Candidate Path Selection

**Turki A. ALGHAMDI**

Thesis submitted for the degree of Doctor of Philosophy in Computing

Supervised by: Prof Mike E. Woodward

Department of Computing  
School of Computing, Informatics and Media  
University of Bradford

2010



**In the Name of Allāh, the Most Gracious, the Most Merciful**

*“Thy Lord hath decreed, that ye worship none save Him, and (that ye show) kindness to parents. If one of them or both of them attain old age with thee, say not "Fie" unto them nor repulse them, but speak unto them a gracious word. (23) And lower unto them the wing of submission through mercy, and say: My Lord! Have mercy on them both as they did care for me when I was little. (24)”*

(The Noble Qur’ân - Sūrah Al-Isra 17, Verses 23-24)

# Abstract

The growing demand on the variety of internet applications requires management of large scale networks by efficient Quality of Service (QoS) routing, which considerably contributes to the QoS architecture. The biggest contemporary drawback in the maintenance and distribution of the global state is the increase in communication overheads. Unbalancing in the network, due to the frequent use of the links assigned to the shortest path retaining most of the network loads is regarded as a major problem for best effort service. Localised QoS routing, where the source nodes use statistics collected locally, is already described in contemporary sources as more advantageous. Scalability, however, is still one of the main concerns of existing localised QoS routing algorithms.

The main aim of this thesis is to present and validate new localised algorithms in order to develop the scalability of QoS routing.

Existing localised routing, Credit Based Routing (CBR) and Proportional Sticky Routing (PSR), use the blocking probability as a factor in selecting the routing paths and work with either credit or flow proportion respectively, which makes impossible having up-to-date information. Therefore our proposed Highest Minimum Bandwidth (HMB) and Highest

Average Bottleneck Bandwidth History (HABBH) algorithms utilise bandwidth as the direct QoS criterion to select routing paths.

We introduce an Integrated Delay Based Routing and Admission Control mechanism. Using this technique Minimum Total Delay (MTD), Low Fraction Failure (LFF) and Low Path Failure (LPF) were compared against the global QoS routing scheme, Dijkstra, and localised High Path Credit (HPC) scheme and showed superior performance. The simulation with the non-uniformly distributed traffic reduced blocking probability of the proposed algorithms.

Therefore, we advocate the algorithms presented in the thesis, as a scalable approach to control large networks. We strongly suggest that bandwidth and mean delay are feasible QoS constraints to select optimal paths by locally collected information. We have demonstrated that a few good candidate paths can be selected to balance the load in the network and minimise communication overhead by applying the disjoint paths method, recalculation of candidate paths set and dynamic paths selection method. Thus, localised QoS routing can be used as a load balancing tool in order to improve the network resource utilization.

A delay and bandwidth combination is one of the future prospects of our work, and the positive results presented in the thesis suggest that further

development of a distributed approach in candidate paths selection may enhance the proposed localised algorithms.

# Acknowledgments

First and foremost I would like to thank Allāh, the most merciful and the most gracious, for His continuous blessings upon me and my family, for giving me the power to believe in my passion and pursue my dreams. I could never have done this without the faith I have in him.

Secondly, I would like to acknowledge and extend my heartfelt gratitude to Umm AlQura University in Mecca for granting me the scholarship to pursue my PhD studies.

I also would like to express the deepest appreciation and gratitude to my supervisor, Prof Mike Woodward, for his abundant help, invaluable assistance and constant support. He generously allowed me to use his knowledge and experience and led me through writing this thesis with sophisticated and delicate guidance advising with the heart of the father and the mind of the scholar.

Last but not least, I would like to give my heartfelt thanks to my parents, my sisters and my brothers.

# **Dedication**

To the two most precious people in my life- my parents, Mr Ali Alghamdi and Mrs Salhah Alghamdi.

Turki Alghamdi

# Table of Contents

<b>Abstract .....</b>	<b>iii</b>
<b>Acknowledgments .....</b>	<b>vi</b>
<b>Dedication .....</b>	<b>vii</b>
<b>Table of Contents .....</b>	<b>viii</b>
<b>List of Figures .....</b>	<b>xiv</b>
<b>Glossary of Abbreviations .....</b>	<b>xvi</b>
<b>List of Published Papers .....</b>	<b>xviii</b>
<b>Chapter 1 Thesis Introduction .....</b>	<b>1</b>
1.1 Background .....	1
1.2 Motivation .....	2
1.3 Thesis Aims and Objectives .....	3
1.4 Thesis contributions .....	4
1.5 A General Outline of the Thesis .....	6
<b>Chapter 2 QoS Routing .....</b>	<b>9</b>
2.1 Introduction .....	9
2.2 Description of QoS Routing .....	10
2.2.1 Notation and Definitions .....	12
2.2.2 Routing process .....	14

2.3 Routing Strategies .....	14
2.3.1 Number of Destinations .....	15
2.3.2 Routing based on the state information.....	16
2.3.3 Routing based on Decision Place .....	18
2.4 Problem Setting .....	23
2.4.1 Metric Distribution.....	24
2.4.2 Path selection algorithm.....	25
2.4.3 Single and multiple constraint routing problems .....	26
2.4.4 NP-complete problem .....	27
2.5 QoS Architecture.....	28
2.5.1 Integrated Services (IntServ).....	29
2.5.2 Differentiated Services (DiffServ) .....	30
2.6 Summary .....	33
<b>Chapter 3 Related Work .....</b>	<b>34</b>
3.1 Introduction .....	34
3.2 Global QoS routing .....	35
3.2.1 Widest-shortest path.....	36
3.2.2 Shortest-widest path .....	37
3.2.3 Shortest-distance path .....	37
3.2.4 QoS extensions to OSPF (QOSPF) [53]: .....	38

3.3 Localised QoS Routing .....	43
3.3.1 Dynamic alternative routing.....	44
3.3.2 Learning automata based routing.....	45
3.3.3 Proportional Sticky Routing.....	46
3.3.4 Localised Credit Based Routing.....	49
3.4 Summary .....	51
<b>Chapter 4 Modelling and Simulation Environment .....</b>	<b>52</b>
4.1 Introduction .....	52
4.2 Simulator design.....	53
4.3 Simulation Environment .....	54
4.4 Simulation Structure.....	55
4.5 Graph Model .....	63
4.5.1 Random Topology.....	64
4.5.2 Regular Topology.....	65
4.5.3 ISP Topology .....	66
4.6 Performance Metrics .....	67
4.7 Simulator Validation .....	68
4.8 Summary .....	71
<b>Chapter 5 New Localised algorithms based on Bandwidth as the QoS metric .....</b>	<b>72</b>
5.1 Introduction .....	72

5.2 Proposed algorithms .....	73
5.2.1 Highest Minimum Bandwidth (HMB) algorithm .....	75
5.2.2 Highest Average Bottleneck Bandwidth History (HABBH) algorithm .....	78
5.3 Performance evaluation .....	<b>Error! Bookmark not defined.</b>
5.4 Methodology .....	81
5.4.1 Simulation environment .....	81
5.4.2 Network topologies .....	81
5.4.3 Simulation Characteristics .....	82
5.4.4 Performance Metrics .....	84
5.4.5 Path selection methods .....	84
5.4.6 Simulation Results.....	86
5.5 Summary .....	103
<b>Chapter 6 Integrated QoS Routing and Call Admission Control Algorithms</b>	
<b>Using a Localised Approach.....</b>	<b>105</b>
6.1 Introduction .....	105
6.2 Admission Control .....	107
6.3 The Proposed Algorithms.....	110
6.3.1 High Path Credits (HPC).....	111
6.3.2 Minimum Total Delay (MTD) .....	114

6.3.3 Low Fraction Failure (LFF) .....	117
6.3.4 Low Path Failure (LPF) .....	120
6.4 Performance evaluation.....	122
6.5 Methodology .....	123
6.5.1 Simulation environment.....	123
6.5.2 Network topologies .....	124
6.5.3 Simulation Characteristics .....	124
6.5.4 Performance Metrics .....	125
6.5.5 Simulation Results .....	126
6.6 Summary .....	141
<b>Chapter 7 A Distributed Approach to Localised QoS Routing .....</b>	<b>143</b>
7.1 Proposed algorithm .....	143
7.2 Distributed High Bandwidth (DHB) algorithm.....	144
7.3 Performance evaluation.....	147
7.4 Methodology .....	147
7.4.1 Performance Metrics .....	148
7.4.2 Simulation Results .....	148
7.5 Summary .....	152
<b>Chapter 8 Conclusions and Future Work.....</b>	<b>154</b>
8.1 Summary and Conclusions .....	154

8.2 Future Work .....	158
<b>References .....</b>	<b>160</b>

# List of Figures

Figure 3.1 Dynamic alternative routing algorithm flow chart [67].....	44
Figure 3.2 PSR Pseudo-code [5]. .....	46
Figure 4.1 Functional components of simulator .....	55
Figure 4.2 a 4-node torus graph.....	66
Figure 4.3 the ISP topology .....	67
Figure 4.4 Simulator validation results. ....	70
Figure 5.1 The pseudo code for the HMB algorithm .....	76
Figure 5.2 The flow chart of the HMB algorithm.....	77
Figure 5.3 The pseudo code for the HABBH algorithm .....	78
Figure 5.4 The flow chart of the HABBH algorithm .....	80
Figure 5.5 Impact of disjoint paths and recalculation in the candidate path set of different algorithms on Rand80 network.....	87
Figure 5.6 Dynamic method performance of HMB algorithm. ....	88
Figure 5.7 Impact of update interval for different algorithms in Torus network .....	90
Figure 5.8 Flow and bandwidth blocking probability in Rand80 .....	91
Figure 5.9 Flow Blocking Probabilities in different topologies .....	95
Figure 5.10 Impact of w parameter in Rand80 topology .....	97

Figure 5.11. Impact of bursty traffic on different algorithms in different network topologies. ....	99
Figure 5.12 Impact of Bursty traffic for different network topologies .....	101
Figure 6.1 Flow chart of Call Admission Control algorithm .....	109
Figure 6.2 Call Admission Control algorithm.....	110
Figure 6.3 The pseudo code for the HPC algorithm.....	112
Figure 6.4 The pseudo code for the MTD algorithm.....	115
Figure 6.5 The pseudo code for the LFF algorithm .....	118
Figure 6.6 The pseudo code for the LPF algorithm.....	121
Figure 6.7 Varying delay constraints in different network topologies ....	129
Figure 6.8 varying arrival rates under tight delay constraint (1.5 time units) in different network topologies. ....	133
Figure 6.9 varying arrival rates under slack delay constraint (3 time units) in different network topologies .....	135
Figure 6.10 Impact of Bursty traffic in irregular and regular network topologies .....	137
Figure 6.11 Impact of non-uniform traffic in regular and irregular topologies .....	139
Figure 7.1 The pseudo code for the DHB algorithm.....	145
Figure 7.2 Flow blocking probability in different network topologies .....	150

# Glossary of Abbreviations

<b>AF</b>	Assured Forwarding
<b>ATM</b>	Asynchronous Transfer Mode
<b>CBR</b>	Credit Based Routing
<b>DAR</b>	Dynamic Alternative Routing
<b>DHB</b>	Distributed High Bandwidth
<b>DiffServ</b>	Differentiated Services
<b>EF</b>	Expedited Forwarding
<b>HABBH</b>	Highest Average Bottleneck Bandwidth History
<b>HMB</b>	Highest Minimum Bandwidth
<b>HPC</b>	High Path Credit
<b>IETF</b>	Internet Engineering Task Force
<b>ICMP</b>	Internet Control Message Protocol
<b>IGMP</b>	Internet Group Management Protocol
<b>IGRP</b>	Interior Gateway Routing Protocol
<b>IntServ</b>	Integrated Services
<b>IP</b>	Internet Protocol
<b>ISP</b>	Internet Service Provider
<b>LFF</b>	Low Fraction Failure
<b>LPF</b>	Low Path Failure
<b>MCP</b>	Multi Constrained Path
<b>MPLS</b>	Multi Protocol Label Switching
<b>MTD</b>	Minimum Total Delay
<b>NED</b>	Network Description Language
<b>OSPF</b>	Open Shortest Path First

<b>PNNI</b>	Private Network to Network Interface
<b>PSR</b>	Proportional Sticky Routing
<b>QoS</b>	Quality of Service
<b>QOSPF</b>	Quality of Service Path First
<b>RIP</b>	Routing Information Protocol
<b>RSVP</b>	Resource Reservation Protocol
<b>SDP</b>	Shortest Distance Path
<b>SLA</b>	Service Level Agreement
<b>SWP</b>	Shortest Widest Path
<b>WSP</b>	Widest Shortest Path

# List of Published Papers

- [1] T. A. AlGhamdi and M. E. Woodward, "Localized QoS routing algorithm based on residual bandwidth", in *Proceedings of the 3rd international conference on New technologies, mobility and security* Cairo, Egypt: IEEE Press, 2009.
- [2] T. A. Al Ghamdi and M. E. Woodward, "Novel algorithms for QoS localized routing in communication networks", in *Proceedings of First Asian Himalayas International Conference on Internet AH-ICI2009*, pp. 1-7, November 2009.
- [3] T. A. Al Ghamdi and M. E. Woodward, "Novel localized QoS routing algorithms", in *Proceedings of the 9th IEEE Malaysia International Conference on Communications*, pp. 199-204, December 2009.
- [4] T. Al Ghamdi and M. E. Woodward, "New algorithm for QoS routing in communication networks", in *Proceedings of 2009 IEEE STUDENT CONFERENCE ON RESEARCH AND DEVELOPMENT*, pp. 77-80, November 2009.
- [5] Turki Al Ghamdi, M.E. Woodward, "Bandwidth as a dominant metric in localized QoS algorithm", in *Proceedings of the 17th Telecommunications forum TELFOR 2009*, pp. 145-148, November 2009.

[6] T. A. AlGhamdi and M. E. Woodward, " QoS Algorithm for Localised Routing Based on Bandwidth as the Dominant Metric for Candidate Path Selection", in *Proceedings of The 10<sup>th</sup> IEEE International Conference on Computer and Information Technology*, June 2010.

[7] T. A. AlGhamdi and M. E. Woodward, " Bandwidth as a Metric for Candidate Path Selection: An Approach to Localized Routing Algorithms in Communication Networks," *International Journal of Simulation- Systems, Science and Technology - IJSSST*, Vol. 10, No. 8, 2010, to appear.

# Chapter 1

## Thesis Introduction

### 1.1 Background

Connectivity is one of the key priorities of routing in the existing internet, which relies on the “best-effort” principle. Most routing protocols work with a single metric, such as hop count or administrative weight. OSPF [1] is a well known example of such protocols and uses the shortest path routing paradigm. The main drawback of using such protocols is a lack of resources which cannot satisfy the requirements of IP telephony or video-on-demand applications. Unbalancing in the network is regarded as a major problem of using the shortest path for best effort service. This happens due to the frequent use of the links assigned to the shortest path which retain most of the network load. Thus, QoS routing [2, 3] is proposed as an essential mechanism. The procedure for selecting the paths in QoS routing is based on the availability of the resources which can satisfy the QoS requirements. Thus, the QoS state is the key factor in the QoS routing, as the path will be selected only if it has sufficient resources to satisfy the QoS requirements. Therefore, the disadvantage of unbalancing the network, which

appears when using the shortest paths in the best effort mechanism, will thus be reduced by using the QoS routing due to the knowledge of the QoS state.

## 1.2 Motivation

A QoS routing mechanism is required to collect the state information and keep it up-to-date. According to the gathered information an optimum path is selected to route a flow. Inherent scalability problems can arise because of the collected information method. These problems are listed below:

- Periodic exchange of global state information can cause prohibitive communication and processing overheads.
- The prohibitive communication and processing overheads entailed by such frequent state updates preclude the possibility of always providing each node with an accurate view of the current network QoS state. Thus, as a result, the network state information gathered at a source node could rapidly result in being out-of-date given the QoS state update time interval is large in relation to the flow dynamics.
- Route flapping can also take place in the global QoS routing as a consequence of exchanging stale state information due to the large update intervals. This scenario can appear when the source nodes direct incoming flows, using stale information, through a link which has in fact has got low utilization. Thus, as a consequence,

utilization of that particular link rapidly increases and vice versa. As a result of the routing in this rapid cyclic manner deficient transmission of the traffic occurs, inducing unstable, corrupted network performance.

The degree of the struggle increases with the size of the network, thus identifying the scalability of QoS routing algorithms as a foremost challenge. Therefore, our motivation is to study existing global QoS routing algorithms troubled by inherent scalability issues in order to develop new QoS routing methods with enhanced scalability.

### **1.3 Thesis Aims and Objectives**

The aims of this thesis are to evaluate the scalability problems and propose QoS routing algorithms leading to development of better scalability of QoS routing. The implementation of localised QoS routing algorithms provided by selecting a path based on information which is collected locally enhances the scalability and reduces the overhead in a network.

Using the metric of interest, such as bandwidth or delay, always allows having up-to-date information, which would not be possible with the existing localised algorithms Credit Based Routing (CBR) [4] or Proportional Sticky Routing (PSR) [5] working with either credit or flow proportion respectively. Any changes in

these metrics due to flows on candidate paths of other nodes can be conveyed back to the source node of all candidate paths involved.

The set aims ought to be met by the subsequent objectives:

- To research several approaches of QoS routing and related scalability problems.
- Integrating a QoS localised approach to develop novel routing algorithms that are based on guaranteeing QoS constraints, low message overhead and efficient resource utilization.
- Developing competent localised QoS routing algorithms with sufficient scalability, employing bandwidth as QoS metric for the path selection method.
- Defining mean delay as a QoS metric and integrating the call admission control principle into localised QoS routing algorithms.
- Selecting a simple and flexible simulation environment in order to assess the efficiency of the offered localised algorithms against popular localised and global QoS algorithms.

## **1.4 Thesis contributions**

This thesis contributes to the following areas:

- Bandwidth as a QoS metric is implemented in two localised QoS routing

algorithms. The proposed algorithms are compared under different traffic loads and network topologies to the existing localised and global routing algorithms and demonstrated better performance.

- Two path selection methods, selection of disjoint paths and recalculation of the set of candidate paths, were used. They improved the performance of the existing localised algorithms without undue increase in complexity.
- A dynamic path selection method applied on localised routing algorithms resulted in a positive effect of their performance.
- An Integrated Delay Based Routing and Admission Control (IDBRAC) mechanism are presented in the thesis and have been implemented as localised algorithms.
- Several localised QoS routing algorithms based on mean delay in order to opt for the optimum path from the candidate path set are presented.
- The High Path Credit (HPC) algorithm, which is the modification of the Credit Based routing algorithm (CBR) [4], is implemented based on mean delay instead of bandwidth. HPC is used as a standard localised algorithm to evaluate our localised QoS routing algorithms based on mean delay.
- Under different scenarios, using both tight and slack delay constraints in various network topologies, our presented localised delay algorithms were compared to the global shortest path algorithm (Dijkstra). The appropriate

simulation illustrated the prevalence of the localised methods over the Dijkstra algorithm in all network topologies, except for the simulations with an impractically small update interval of link state for the global schemes.

- The thesis also includes a Distributed High Bandwidth (DHB) algorithm which differs from previous localised routing algorithms by using a distributed routing approach instead of a source routing approach. A comparison of the new strategy, DHB, with the existing localised algorithm, CBR, demonstrate the superiority of DHB.

It should be noted that the localised routing is intended to relate to routing within a sub-network or backbone network but not an entire internet.

## 1.5 A General Outline of the Thesis

The structure of this thesis finds the content divided into eight chapters.

**Chapter 1: Thesis Introduction** - introduces the reader into background of the research sphere. It gives a straight to the point introduction of the topics to be covered in the research area and highlights the motivation to our work, aims, objectives and work contribution. The last section provides a brief structure of the thesis.

**Chapter 2: QoS Routing** - is concerned with a general overview of QoS and examines specific QoS routing strategies. The chapter also provides a speculation about QoS routing setbacks.

**Chapter 3: Related Work** - Relevant work linked to the area of unicast QoS routing based on the shortest path algorithms, global routing algorithms and localised routing algorithms is presented in chapter three.

**Chapter 4: Modelling and Simulation Environment** – This chapter conveys a simulation model of QoS routing, which allows us to evaluate the functioning of the proposed algorithms in the two consequent chapters. In addition, this simulation assists us to contrast between the suggested algorithms and other current algorithms subjected to varying configurations. Simulator design and validation also contribute to this chapter. The characteristics of employed types of network topologies, the algorithm parameters and the performance metrics used in the performance evaluation are also presented.

**Chapter 5: New Localised Algorithms Based on Bandwidth as the QoS Metric** – Two new localised QoS routing algorithms based on bandwidth as the QoS metric (HMB and HABBH) are illustrated. This is carried out by an extensive simulation evaluation of the proposed algorithms against the other localised routing CBR and the global QoS routing WSP.

**Chapter 6: An Integrated QoS Routing and Call Admission Control**

**Algorithm Using Localised Approach** - QoS routing algorithms using an Integrated Delay Based Routing and Admission Control (IDBRAC) mechanism are introduced in this chapter. The localised Credit Based Routing (CBR) is modified to use mean delay instead of bandwidth and produces the (HPC) algorithm. A comprehensive description of all algorithms is also provided. An experimental study tested the difference in performance between the proposed algorithms against the global shortest path algorithm (Dijkstra).

**Chapter 7: A Distributed Approach to Localised QoS Routing** - This chapter

investigates a distributed approach with localised routing apart from source routing approach previously described in chapters 5 and 6. The Distributed Highest Bandwidth (DHB) algorithm is a new mechanism compared under extensive simulation to the existing localised algorithm (CBR).

**Chapter 8: Conclusions and Future Work** - In this final chapter, the research is

summarised and the main conclusions are drawn. The chapter concludes with suggestions for further enhancements.

# Chapter 2

## QoS Routing

This chapter provides a general literature review on QoS routing with a focus mainly on examining specific QoS routing strategies.

### 2.1 Introduction

QoS for network communication is a mechanism that ensures high quality communication in the network application. The traditional concepts of network quality in network traffic are treated equally where there is no guarantee of best-effort service delivery, and bandwidth intensive applications can result in poor and unacceptable performance in all applications. QoS routing relates to routing algorithms capable of identifying paths to satisfy specified constraints. The protocols in achieving the QoS routing implementation must also achieve efficient utilisation of resources and must be able to support traffic using integrated services and achieve maximum flows subject to end-to-end constraints. For effective network path selection, QoS routing should support multiple paths and be able to shift from one path to another upon finding a better path. The dynamic selection of a feasible choice, and optimisation of resource utilisation are some of

the requirements of a QoS routing mechanism [3, 6, 7]. The next section provides an overview of QoS routing in communication networks.

## **2.2 Description of QoS Routing**

QoS routing has generated much interest among scholars [8-10] and has recently received renewed attention in the domain of both wired networks [11-14] and wireless networks [15-17] because of its features in the interconnection of both types of network and its support to the connection to internet connectivity. Gudrin, Kamat, and Tripathi [18] point out that QoS routing is the process of selecting a path used by the packets based on the requirements of the flow. The motivation of using path selection is to improve the service received by users for all network efficiencies. However, computational cost and protocol overhead can lead to increase in computation complexity and this can become a primary inhibitor to QoS deployment. Efficient gains in QoS routing can be achieved by obtaining simpler routing protocols, and QoS routing can be enhanced by suitable network topologies. QoS routing is a tool to improve the flows of networking, and achieving sufficient resources to meet a flow's requirement. Meeting flow requirements in the form of bandwidth, and network resources means QoS routing has become an essential tool to guarantee network flow, which enables a network operator to identify a path for every new flow, and meet network requirements.

The network requirements in the form of bandwidth are suitable to find a path to a destination. In identifying path selection, there is a need to find paths with sufficient resources to accommodate requirements for path flow. It should be noted that each criteria of QoS routing is capable of satisfying the path selection process if a path is capable of minimising the amount of resources being consumed. Feasible paths, capable of satisfying flow requirements, depend on the accuracy of information in the network resources [19, 20].

Hu, and Johnson dispute that QoS is very important in wired networks, where over-provisioning can reduce the need for sophisticated QoS techniques. Using a QoS routing protocol needs a careful analysis of routing paths for wireless network applications though. Routing selection can enhance performance of network traffic. Thus, in ad hoc networks, capacity and connectivity are quite dynamic [21]. With the argument presented by Shaikh [22], QoS routing can satisfy requirements for application performances and optimise network usage, providing that paths are selected based on connection traffic parameters. There is significant bandwidth and processing overhead that can be imposed by QoS routing mechanisms since they can impose significant bandwidth and processing load on the network [22-24].

As described by Ouferrhat, and Mellouk [25], QoS routing algorithms involve the process of selecting, discovering and maintaining pathways from one node to

another to satisfy a specific service requirement. By using these routing paths, the networks deliver flows of packets. Performance of QoS routing thus largely depends on the traffic patterns and the network topology. In network communication, the effectiveness of routing protocols has an impact on routing decisions. A routing protocol generally consists of the task of capturing the state of a network and its available resources for dissemination of information in the network. However, a routing failure occurs if the source selects a path that cannot support a new connection, and a set of failures can incur extra overhead and waste of resources along the routing path [25-27]. Although QoS routing has become a topic of considerable interest with the increase in popularity of real-time multimedia applications, there has also been an evolution of complicated problems. Although evidence has revealed QoS routing can provide an increase in network utilisation, there is still an argument that QoS routing can be costly to maintain because of the increase in the routing protocol overheads [18].

### 2.2.1 Notation and Definitions

Before describing the routing framework, some definitions are in order. To begin, a network can be illustrated as a directed graph  $G$ , which consists of a pair  $(N, L)$ , where  $N$  and  $L$  indicate the set of nodes and set of links, respectively. A link  $l=(x, y)$  consists of two nodes  $x, y \in N$ , and each link holds  $m$  values of its metric, with

one for each link, symbolised as  $m$  weights:  $h_1(x, y), h_2(x, y), \dots, h_m(x, y)$  where  $h_i(x, y) > 0 \forall (x, y) \in L$ . Path  $p$  from node  $s$  to node  $d$  in  $G$  is indicated by  $p = x_{s=0} \rightarrow x_1 \rightarrow x_2 \rightarrow \dots \rightarrow x_{d=l}$  given that  $(x_i, y_{i+1}) \in L$  for all  $1 \leq i \leq l-1$ .

A path  $p$  is required to be located by a QoS routing algorithm, which fulfils the QoS conditions of a flow. These conditions are quantitatively assessed by employing chosen metrics to ensure that the requirements are illustrated as either one or a permutation of metrics. In QoS routing, the routing algorithm generates a path that must satisfy the QoS conditions and a set of mathematical specifications that decide how the QoS conditions must be executed. The constraints can be represented as a set  $C$  of  $m$  constants ( $c1, c2, \dots, cm$ ), where  $c1$  represents a constraint for metric  $h1$ ,  $c2$  represents a constraint for metric  $h2$  and so on.

Where a path is able to meet all the given constraints it is referred to as a *feasible path* and there can be several such paths. In some instances the key function is to determine the optimum feasible path based on an optimization metric. For instance, determining the least cost path where the mean delay is predetermined and assigned a specified value. In this case, the optimization goal is represented by the cost and the mean delay is a constraint. These types of problems are called QoS routing with optimization.

Concave metric constraints are generated from a single link along a network path and thus are also referred to as *link constraints*. Another form of constraint is

known as a *path constraint* which is determined by utilising all links that make up the given path. A concave metric also comprises bandwidth which is conceded as one of the most utilized metrics. In addition, additive metrics comprise delay, delay jitter, cost, and hop count. Types of metrics will be discussed in detail in section (2.4.2) in this chapter.

### **2.2.2 Routing process**

As mentioned earlier, routing is the process of selecting paths in a network along which to send network traffic. Basically, the routing process comprises two functions: first, obtain information on the state of a network and keep it up to date, second, select a path in a network along which to transport network data. The first function conveys to each node the state of the network, while the second function is executed given the information obtained in the first task. The network path is computed by a *routing algorithm* and computing the optimum path is dependent on the procedure for gathering information, and the information storage system.

We will now discuss the first of the routing functions while discussing the second function at a later stage.

## **2.3 Routing Strategies**

Establishing a feasible path or routing methodology can be complicated due to traffic requirements and functions to be carried out. Primarily the function of

routing is to gather state information and maintain its accuracy. Routing also includes the task of determining the best route path which minimises network costs. Routing methods can be categorised based upon the way in which the state information is maintained, and the methods of determining feasible paths [2].

From this point of view several classifications have been proposed for routing methods.

### **2.3.1 Number of Destinations**

Routing protocols can be classified according to the number of nodes that participate in the routing event. When there is a single node as a sender and a single node as a recipient, the routing event is called unicast routing. In case of one node a sender and a group of nodes as receivers, the routing event is called multicast routing. Flood routing or broadcast routing is the routing event in which one node acts as a sender and all other nodes in the network act as receivers.

#### **Unicast Routing**

Unicast routing is the process for forwarding network traffic from a node to another node in the network. The network traffic is addressed to a unique address. Therefore, the network traffic contains the address of the destination node and the source will forward network traffic so addressed to the destination node through intermediate nodes.

## **Multicast routing**

Multicast routing enables a node to send network traffic to a subset of the other network nodes. Multicast routing is carried out in two steps. The first step is to find a tree that connects all nodes in a multicast group and then send the network traffic to this group.

## **Flood Routing**

In flood routing or broadcast routing a source node sends network traffic to all other nodes in the network.

### **2.3.2 Routing based on the state information**

In this routing category, there are three techniques of gathering the state information, which we will now discuss.

#### **Local state**

Each node in the network is required to ensure that the routing algorithm has the correct and latest state information in order for it to compute the network path along which to send network traffic. State information comprises bandwidth, network delay, hop count, and other data of relevance to the routing process [28].

## Global state

The global state is obtained by a process of combination all nodes' local states. In other words, nodes maintain a global state of the network by sending and receiving local state information with other nodes. This is performed in two ways [29] :

- *Distance Vector Protocol*

A distance vector routing protocol employs the Bellman-Ford algorithm [29, 30]. The protocol calculates the path and distance to all nodes across a network where each router in the network computes a routing table (vector based), which provides the optimum path to each destination. The routing tables are maintained by sending each router's information to its immediate neighbour. When the information is received, the routing table is updated to reflect any new information and allowing nodes to realize the optimum hop count to the destination.

RIP [31] adopts hop count to send network traffic to a destination node. IGRP [32] uses several multiple metrics for each route, including delay, and available bandwidth.

- *Link State Protocol*

OSPF and IS-IS [33] are both examples of link-state routing protocols. Each router discovers its neighbours, measures the required metrics and sends this

information to all other routers in the network. 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 global state**

This state represents the hierarchical structure of the network. Hierarchical models are often used to reduce the size of the global state by having each node store more detailed information about nodes that are close to it and less information about the nodes far from it.

### **2.3.3 Routing based on Decision Place**

Source routing, distributed routing and hierarchical routing are all methods of routing in which computing the optimum path can be executed based on the decision location.

#### **Source Routing**

Source routing is a procedure that can be utilised to indicate the path that a packet should adopt through the network. In source routing, the route via the network is specified by the sender of the data packet. The assumption is that the sender of the packet has an understanding of the framework of the network and can thus

indicate the optimum path for the packet or flow. The process of maintaining routing in a network is by issuing control messages to the chosen path by connecting intermediate and subsequent nodes. Both link state protocol [1] and distance vector protocol [34] are utilised to relay global state information between nodes in the source routing method. It is suggested by Flich et al that source routing is observed to enhance the functioning of irregular networks for the average rate of successful message delivery [35]. The simplicity of source routing is largely due to converting a distributed problem into one involving a centralised state.

Also, source routing can be categorised as on-demand computation [9] and pre-computation [36, 37]. In the former, a QoS route is determined on a per-request basis, which results in an increase in the computational overhead. With the pre-computation method, QoS paths are calculated in an asynchronous way with request arrivals. These paths are implemented to create multiple network traffic which has the same destination, therefore reducing the computational overhead or complexity.

The key benefit of source routing lies in its simplicity, since the source node will compute the complete path. Thus routers function merely as storage and forwarding applications. Also, source routing offers the advantage of eliminating the burden of deadlock from the routing algorithm whereby packets continually

wait for resources in a cyclic manner. Another advantage afforded by source routing is that it is feasible to implement several algorithms in the same network given that routes are determined on a local basis at each node. As a result, this reduces, to a great extent, the overhead incurred from large networks and allows the network administrator to easily deploy and determine the viability of the latest routing algorithms. It must also be noted that source routing incurs potential limitations, such as QoS routing failure.

### **Distributed Routing**

As pointed out by Lee et al [38], distributed routing strategies have increased in recent years due to increase in demand for transaction processing rates, and development of multi processors or locally distributed systems. The data sharing approach in the network process, and hybrid data sharing database systems have contributed to the increase in the demand for distributed sharing of routing [38, 39].

In the case of distributed routing algorithms, routers produce decisions at every intermediate node from source to destination. The routing path is computed at each hop in the route path at each and every node.

In the case of distributed routing algorithms, where each node is required to maintain the global state, such an algorithm suffers from the issue of inaccurate state information and high protocol costs, as is the case with source routing.

However, distributed routing algorithms that are responsible for maintaining only the local state information do not inherit these constraints [6, 35, 40].

Internet routing protocols, including RIP (Routing Information Protocol) [31], adopt hop count as choice of metric. The overheads incurred via the computation are less because there is a distribution of such overheads amongst intermediate nodes. Distributed routing protocols are more scalable compared with source routing protocols. The benefits of distributing routing are that the route computations are spread amongst several nodes. This means a reduction in the routing response time and increases the scalability of the algorithm. It is also viable to determine several optimum paths concurrently. However, a constraint associated with distributed routing is inaccurate information relating to the global state that is maintained by each node. This can result in an increase in routing failures and waste of resources.

### **Hierarchical Routing**

Intricate and more difficult routing issues in larger networks necessitate reducing the information that needs to be handled by using a hierarchical network in which each position of the network is tasked with its own routing [41]. Even though the hierarchical routing techniques have problems in implementation, these schemes should be generally used to reduce the overheads in wired networks [42, 43]. 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 meets the requirements of both source and distributed routing. As soon as connection requests arrive, the computing of feasible paths is done by using source routing algorithms. Distributed routing mechanisms are also used in a hierarchical structure by distributing computation of paths over many nodes [22, 44, 45].

The common problem of finding users in a mobile network is becoming a prominent issue as this can result in bandwidth issues. Such problems can be tackled by Hierarchical Routing Protocols, as pointed out by Pei and et al [46].

The researchers Lauder, Kummerfeld, and Fekete [45] demonstrate hierarchical routing as a network of machines which are assigned their own name in a systematic manner and all the names are illustrated in the form of a tree based structure. In order to achieve an optimum network performance where there is the deployment of a hierarchical routing system, there is a requirement to represent the network in a hierarchical form to decrease the overheads [47].

To obtain aggregate global state information in a hierarchal network, nodes are clustered into groups. Also nodes can be fragmented further to produce a multi-layered hierarchy, this method causes each node to ensure aggregate global state

information. At the same time, every node within a group holds aggregate information and detailed state information relating to other groups and its own cluster. With networks of very large size performance of hierarchical algorithms is noticeable and produces better results. It is well known that the hierarchical approach gives a logarithmic reduction in network state information [48]. Therefore, the complexity is decreased in the large networks [49]. One of the most useful benefits of hierarchical routing is the fact that every node holds correct routing computations relating to the global states [2, 6, 35]. In comparison, they exhibit the problem of inaccurate state information as a consequence of the aggregation, which increases as the number of aggregated levels increases [50]. Private Network-Network Interface, PNNI [51] is a common example of a hierarchical routing protocol as deployed in ATM networks.

## **2.4 Problem Setting**

The key function of QoS routing is to determine the optimum path to transport packets to their destination. This is performed by evaluating all the available information about the network state [3]. Another aim of QoS routing is to give better use of network resources. QoS routing algorithms should solve both issues: selection of paths and metric distribution mechanisms. Single and multiple constraint routing problems are also presented in this section.

### 2.4.1 Metric Distribution

The network state can be illustrated by several properties, or metrics, of a route. Such metrics include path reliability, path bandwidth, and latency (delay and jitter). The aim of traffic characterization is to acquire an understanding of the properties of the traffic. The method(s) employed for traffic characterization determines the way in which traffic requirements are illustrated. For example, in the Integrated Services framework, it can be achieved by utilising the QoS parameters associated with every data flow throughout the resource reservation process [52, 53].

The metrics relating to the network state, i.e. state information, have to be conveyed to some or all the routers in the network and this task is performed more generally than in a conventional routing model. This is because it is required to illustrate the continuous changes taking place across the network. However, it must be noted that if the network state information is maintained too frequently then this consumes more network bandwidth, and which is not an ideal scenario.

Thus, in such a scenario it is best practice to achieve a balance between the accuracy of state information and the cost that updating this incurs. There are several methods to tackle this problem including the distribution of quantified values in place of instant values. In relation to these quantified values, triggers can

be employed as a mechanism for controlling the release of updates and timers used to impose a lower fixed time period between the release of updates [18].

Several issues arise from a lower frequency of communicating network state information including unreliable state information, delays in transporting network data, the exploitation of approximation, the effect of the metric measurement mechanism recruited and information aggregation in hierarchical systems. Numerous studies have investigated the consequences of unreliable routing information on the performance of networks in routing data and the methods to resolve such issues [19, 39, 54, 55].

### **2.4.2 Path selection algorithm**

The path selection algorithm generates its intricacies given that the traffic of data has various QoS requirements, and thus the path selection algorithm must determine paths that fulfil specified requirements and constraints. This presents difficulties due to the calculations involved, given the choice of metric employed.

The value of a chosen metric, based on its value in each hop, depends on the type of metric. The three most widely employed forms of metrics, also referred to as the composition rules of the metrics, are as follows [8, 56]:

- **Additive metric:** the value of this metric over a path is the sum of the values associated with each hop. Examples of additive metrics delay and hop count.

$$h(p) = \sum_{i=0}^{m-1} h(x_i, x_{i+1}).$$

- **Multiplicative metric:** the value of the metric over a path is the product of its values in each hop. Packet loss is an example of a multiplicative constraint.

$$h(p) = \prod_{i=0}^{m-1} h(x_i, x_{i+1}).$$

- **Concave metric:** the value of concave metric over a path corresponds to the minimum value observed in all hops of that path. For example, bandwidth is a concave metric.

$$h(p) = \min [h(x_i, x_{i+1})] \quad \text{where } i = 0, \dots, m-1.$$

### 2.4.3 Single and multiple constraint routing problems

Single constraint routing issues can be categorized in to four distinct forms:

**Path optimization problem:** determines the least cost routing path by employing the shortest path algorithm

**Path constraint problem:** determines the least cost path based on the total cost of the individual hops along a path. This is implemented using the standard shortest path algorithm.

**Link optimization problem:** determines the least cost path based on the minimum cost of individual links in a path. This problem can be resolved by implementing a variant of Dijkstra's algorithm [57, 58] or Bellman-Ford's algorithm [29, 30].

**Link constraint problem:** This problem can be resolved by selecting a path where the link bandwidths equate to or are higher than a desired bandwidth prerequisite. This can also be approached by pruning links; the links which have a bandwidth lower than the specified requirement are trimmed (hence pruning) and then computing the shortest path in the “pruned” links.

Multiple constraint routing issues, which consist of a single additive and single non-additive metric can be resolved under polynomial time by employing the conventional shortest path routing protocol.

#### **2.4.4 NP-complete problem**

This problem is also known as the multi-constrained path problem (MCP). A distinct occurrence of the problem is when the issue comprises at least two additive metrics [8, 59]. In such an instance the problem is referred to as NP-complete, which means there is no efficient algorithm that can solve this problem exactly and heuristic based algorithms have to be used. This problem continues to be explored extensively in determining a feasible path.

## 2.5 QoS Architecture

IP networks are constructed on the concept of optimum effort networking. This does not make any guaranteed provisions in relation to delivery or how fast and reliable the data is. This particular model is appropriate for a large number of applications and it functions well for nearly all applications given a low network load.

Two key main determinants that necessitate further capability of QoS guarantees; A large number of web applications function in real time and are related to other multimedia data, which have increased service requirements; The other is that the usage of the Internet and its related technologies is continually increasing. Even though network infrastructure is maintained, it is not always the case that network facilities and capabilities will satisfy network demand. To tackle this situation, the IETF has produced two architectures to ensure QoS based handling of data flows in IP networks. The following section details and contrasts the two architectures suggested for QoS; Integrated Services (IntServ) [60]; and Differentiated Services (DiffServ) [61].

IntServ architecture aims to reserve resources across a network via the implementation of the Resource Reservation Protocol (RSVP) [73]. This protocol is not responsible for transporting data across a network, but is an internet

protocol, similar to ICMP, or IGMP, and the implementation of RSVP does yield network costs.

Another approach is DiffServ, also referred to as differentiated services. In this approach, packets are classified depending on the type of service required and then, according to these classifications, routing and switching devices will adopt several methods to ensure that QoS is satisfied according to the classification determined by the DiffServ architecture.

### **2.5.1 Integrated Services (IntServ)**

As mentioned above, the IntServ architecture, developed by the Internet Engineering Task Force (IETF), is tasked with reserving resources on a network. The premise underlying this architecture is to complement the current Internet infrastructure by providing additional services to be deployed across a network. With the IntServ architecture, QoS indicates the type of service being conveyed over the network, identified by features such as bandwidth, and network latency. Using the IntServ approach provides flexibility as nodes in the network have the ability to manage data packets and ensure they satisfy the necessary constraints. A network node which is IntServ compliant has the capability of providing several IntServ services. Also, an IntServ aware node is responsible for providing assistance to the interfaces required by the IntServ architecture, but it is not

afforded with the ability to provision the service itself; an IntServ node does comprehend the variables of the required service.

A significant feature and makeup of the IntServ architecture is the efficient management of network resources. Thus, accordingly, to ensure optimum resource use, the network data has to ensure it satisfies certain constraints; The IntServ approach employs the RSVP protocol to facilitate the function of indicating the required resources.

To satisfy the QoS, RSVP [62] maintains network resources over a routing path in a sequential process. The initial routing device in the network will convey to the next device that there is a request for resource reservation. This is performed until the destination node is arrived at with the same process then being performed in the reverse direction. IntServ services pertain to the *guaranteed service*. This is most closely associated to virtual circuits, while on the contrary the Controlled Load Service is equivalent to the best effort service with no network traffic load.

### **2.5.2 Differentiated Services (DiffServ)**

Another approach is DiffServ, also referred to as differentiated services. In this approach, packets are classified depending on the type of service required and then, according to these classifications, routing and switching devices will adopt

several methods to ensure that QoS is satisfied according to the classification determined by the DiffServ architecture.

As described in section 2.5 earlier, with the Differentiated Services (DiffServ) approach [61] packets are classified dependent upon the service required. The classes of packets which require more extensive resources receive preferential treatment by the DiffServ supporting network. The drawbacks identified with the IntServ approach were the impetus for the development of the DiffServ architecture. More specifically, IntServ was shown to be non scalable in large networks. Performance did not increase in proportion to the capacity added to the network resources. The DiffServ process is designed on the utilization of a section pertaining to the IP header known as DS. This is part of the Type Of Service (TOS) segment of the IPv4 header, and the Traffic class section of the IPv6 header [63]. Clients that request consumption of the DiffServ architecture assign the DS section with a certain value, where this value indicates the Per Hop Behaviour (PHB) for the client's data packets. The viable DS values are stated in a Service Level Agreement (SLA) for supplier and client. The SLA stipulates the requirements relating to bandwidth, transmission, rejection priority, and queue priority. Expedited forwarding [64] and the assured forwarding [65] are two behaviours pertaining to the DiffServ architecture. The difference between the two lies in the packet forwarding process [63]:

**Expedited Forwarding (EF):** The Expedited Forwarding (EF) model is employed for the provisioning of resources to latent/delayed traffic, without compromising QoS. Features of EF include low delay, low loss and low jitter. EF is generally implemented for voice over IP media traffic by telecommunication companies [64].

**Assured Forwarding (AF):** Assured forwarding provides guarantee of delivery providing it is subject to subscribed rates. If traffic goes over the subscribed rates it is rejected outright, but the performance will be compromised [65].

AF prioritises traffic when congestion prevails in the network, using classes with higher prioritization given to the higher class. Several methods are used to manage traffic: Classification, marking, metering, and shaping.

While these methods are generally applied in the above order, metering can however take precedence over marking. These four methods are not required to be deployed for all the routing devices in a network, but rather only need to be implemented at the boundary level routers. This particular characteristic of DiffServ resolves issues relating to the scalability problem inherent with IntServ, given that the core routing devices that can negotiate larger flows are not required to implement the four methods on traffic flows.

## 2.6 Summary

In this second chapter we have defined QoS routing strategies from the point of view of three different categories, such as number of sources and destinations, state of the information and decision place. Number of sources and destinations in its turn is represented by flood, multicast and unicast routing, this latter is to be used in our work.

According to the available sources, there are three techniques of the information gathering referred to as global, aggregated global and local state information gathering. The algorithms presented later in this thesis will make use of the local state approach.

The importance of the decision place category arises from the decision location computing the optimum path. From the source, distributed and hierarchical methods, the source routing approach and also the distributed routing approach are to be used in our algorithms.

# Chapter 3

## Related Work

### 3.1 Introduction

QoS routing is required to perform two key functions to fulfil connection requirements: Path computation, and gathering of state information. The QoS routing protocols are tasked with the gathering and managing state information and ensuring that it is as accurate and reliable as possible in real time, without too much lag in maintaining its accuracy. Dependent on the approaches for maintaining state information and computing viable paths, there are three routing methods as described in the preceding chapter: source, distributed, and hierarchical.

The constraints associated with QoS routing have been the subject of focus in many settings, with various suggestions for the provision of QoS routing. These suggestions do vary as to the location of the selection of the path (i.e. either source or hop-by-hop), how the network state information is collected either by global updates or local observations, and which path is chosen, widest or shortest for example. A study of several QoS routing models is documented in [2, 66-68]. Nevertheless, there are issues pertaining to QoS routing algorithms in terms of

how complex, optimum, and scalable, they are [69]. In general, we can separate routing schemes into *global QoS routing* schemes based on global link state updates, and *localised QoS routing* schemes based on local path state recordings. The subsequent sections describe these QoS routing models in more depth.

### 3.2 Global QoS routing

Global QoS routing models [3, 8, 9, 18, 52, 53] require regular update of *link QoS state* information between network nodes in order to capture a *global illustration of the network QoS state* and keep this up to date. Based on this *current* global view of the network state, a source node dynamically calculates the “best” viable route for a flow which starts from it to a destination node. This particular approach to QoS routing is known as *global QoS routing*. The suggested global QoS routing models are different in terms of path selection method and the network state update invoking procedures.

Path selection algorithms are concerned with the key exchange of the returns from minimising the resource consumption and the balance of network load. The resource consumption by a flow can be minimised by choosing the shortest route with the heaviest load. The network load can achieve a state of equilibrium by selecting the path with the lowest load but if this is a long path it could thus use more resources. Many path selection algorithms have been introduced with the

challenge to limit the hop count and stabilise the load in different ways. These are widest-shortest path (*wsp*) [18, 53], shortest-widest path (*swp*) [8], and shortest-distance path (*sdp*) [9, 70]. These algorithms try to choose a *feasible* path, where a path is deemed to be *feasible* if its *bottleneck bandwidth* (tightest availability of bandwidth along the path) is more than or equal to the requested bandwidth. The models described above vary in terms of choosing a feasible path in the context of several options described as follows.

### **3.2.1 Widest-shortest path**

The WSP Algorithm [53] selects the least distance feasible path with the lowest hop count between paths that fulfill the bandwidth restrictions. Where there are several paths with an equal hop count then the path with the maximum available bandwidth is chosen. The widest path is selected when there are several paths with the same distance. The WSP algorithm achieves an efficient consumption of resources by opting for the shortest path to arrive at the destination. Nevertheless, WSP suffers from the same issues relating to a Minimum Hop Algorithm as the path computation is executed among the shortest feasible paths which are consumed fully before switching to alternative viable paths.

### 3.2.2 Shortest-widest path

Shortest-widest path Algorithm (SWP) [8] locates the widest feasible path which has the largest available bandwidth. If there are several paths having equal width, the lowest distance path is selected. In SWP, Dijkstra's algorithm is implemented twice so as to determine the more ideal feasible route. The hop count or delay metric is only utilized if there are multiple paths containing equal bandwidth congestion. The SWP algorithm has a tendency to opt for the widest path thus allowing it to maintain the distribution of network traffic in a more efficient manner and avoid possible increased latency pertaining to short but heavily used paths.

### 3.2.3 Shortest-distance path

In this algorithm, a feasible path which contains the shortest distance is chosen.

The distance function for a path  $p$  is defined by:  $disp(p) = \sum_{i \in p} \frac{1}{h(i)}$ , where  $h(i)$

indicates the bandwidth on offer on a link  $i$  along path  $p$ . The SDP algorithm [70] has a preference for paths with the lowest loads and factors in hop counts [9]. Also, the SDP algorithm is subject to modifications to resolve the least cost by adopting the function, as follows:

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

range of path algorithms.

By varying the variable  $k$ , in the function above, this leads to a wide array of selection among the shortest path ( $k=0$ ) and widest path ( $k \rightarrow \infty$ ) [70].

### 3.2.4 QoS extensions to OSPF (QOSPF) [53]:

Guerin et al. [53] suggested processes and arrangements to expand the OSPF intra-domain routing protocol. The aim is to establish a structure and viable schemes of QoS routing functionalities resulting in only the minimalist alterations to current routing structures. The authors hold the assumption that an existing link state database is present, providing a list of the availability on every bandwidth link. Several modified forms of the standard widest-shortest path are presented, each with varying calculative and storage requirements. We introduce three modified forms.

To start with, the first algorithm allows each source node to perform a pre-computation regarding the minimum hop path containing the maximum bandwidth for all viable destinations. This is executed by implementing the Bellman-Ford algorithm, whereby on iteration  $h$  it establishes the most favourable or optimal path (containing the maximum bandwidth) from source and every destination among paths of at most  $h$  hops. When iteration  $h$  is computed, then the

$(n, h)$  listing of the routing table is constructed. This provides the maximum bandwidth available along a path of at most  $h$  hops from source to the destination node  $n$ .

Analysis of the deployment of this algorithm on the routing physical device can be consulted in [68]. On demand routing is executed by the second algorithm and utilises an accepted Dijkstra shortest path computation on a graph. All links emanating from the graph which do not contain sufficient bandwidth are then subjected to the pruning process.

The second algorithm, describes functions without the need for a QoS routing table. It utilises the recent updated link information on each state. On the other hand, the first algorithm requires less computation as a result of its ability to pre-compute.

The third algorithm performs a pre-computation on the minimum hop path by employing an accepted Dijkstra computation for every destination by approximating a continuous range of bandwidth values, where the spread of bandwidth request occurrences corresponds to a limited number of classes. The drawback of such a method is lower accuracy in the produced path because of quantization.

Every network node in a global QoS routing method produces link state updates by relaying to the other nodes the latest up to date state of the links connected to

it. Several update procedures are viable and they vary in terms of when the update is invoked and also as to the type of information held in it. Of the majority of the methods suggested thus far [39, 71, 72] the transfer of information for updates depends on the availability of bandwidth at that moment in time. On the other hand, other schemes, such as PSR [5], reciprocate information concerning the probability of the availability of the requested bandwidth.

Existing internet routing protocols, for example, OSPF, disseminate connection information across the network in order that “shortest” routes can be chosen. Managing and ensuring errorless network connection data generally involves a very small number of routing updates because network connection updates are not frequent. Nevertheless, to maintain an errorless network QoS state needs the regular reciprocation of information among network nodes due to the fact that the network resources which are available change in response to every flow arrival and departure. The prohibitive communication and processing overheads entailed by such frequent QoS state updates preclude the possibility of always providing each node with an accurate view of the current network QoS state. Thus, as a result, the network QoS state information gathered at a source node could rapidly result in being out-of-date given the QoS state update time interval is large in relation to the flow dynamics. In such a situation, as described above, the reciprocation of QoS state information between network nodes to keep the QoS

state up to date is more than what is possible without increasing a prohibitive communication overhead. In addition, selecting the path on the basis of a deterministic algorithm, for example, Dijkstra's shortest path algorithm, whereby QoS state information is regarded as correct, doesn't appear to be prudent. The best path selection, factored upon information containing errors, can possibly result in instability; once a QoS state update has taken place, several source nodes select routes containing shared links due to the bandwidth which they perceive to be available, thus resulting in an over consumption of such links. Following the subsequent QoS state update, source nodes will not select paths containing these shared links, leading to them being under-utilised. Such wavering behaviour may result in a considerable degradation on system performance given that QoS state update lag is large. As a result of these constraints, in the situation when the QoS update lag is large in relation to the flow dynamics, the performance of global QoS routing methods deteriorates sharply [73].

Many methods have been suggested to tackle such inaccuracies that are bound to occur in the information availability in the path selection procedure. One method [18] distinguishes the inaccuracy into systematic and random based on the form of update mechanism utilised. Where a change type trigger is used, it is feasible to deduce the spread of actual link metric value on the basis of its most recent broadcast value. Regular types of inaccuracies of this form can be dealt

with by use of a path selection algorithm and thus select a route which is probable to having the requisite resources. In [19] a path selection algorithm is propositioned in order to determine the most reliable path on the basis that the information pertaining to the probability  $p_l(x)$  that a link  $l$  can handle a flow requiring  $x$  units of bandwidth, is known by the source node. This has been the subject of additional experiments conducted by the same researchers in [55] on safety-based routing, where safety, i.e.,  $p_l(x)$ , of a link  $l$  for a bandwidth  $x$  is deduced from its last broadcast bandwidth availability value on the assumption a change type update invoking procedure is employed. Even though the schemes mentioned above minimise the level of inaccuracy on the performance of path selection, the schemes perform satisfactorily in some instances or they lead to extra overhead pertaining to the key routing device. In order to produce simple scalable QoS routing, it is required to construct schemes which will provide sound performance, but not by generating further complexities at the essential routing devices and further communication overheads on network than existing routing protocols. An option to global QoS routing is localised QoS routing, where there is no requirement for global QoS state information reciprocation between network nodes, but rather network source nodes deduce the QoS state on the basis of flow blocking data gathered locally, and carry out flow routing by

utilising the localised impression of the network QoS state. Several localised QoS routing methods are discussed in what follows.

### **3.3 Localised QoS Routing**

In QoS routing, when the source node receives a request with detailed QoS needs, it will determine a route to the destination node dependent on the node's perception of the resources currently available. Following this, a connectivity request is sent in order to reserve resources at every node in the route. It can be the case that there exist insufficient resources on the selected path. This is due to: state routing information at the source node or due to modifications in the state of the network when the connection is being made. Under such a scenario there is a rejection to the request and there is a block to the flow.

Localised QoS routing methods try to deduce the state of the network using this flow blocking data and compute path selection using this local information. There have been numerous localised dynamic routing methods proposed in the environment of telephony networks and we first look at two schemes which are built on sticky routing and learning automata that utilise feedback information in relation to flow acceptance or rejection in regards to routing flows at a subsequent point in time. In localised QoS routing there is the requirement that every source has to initially establish a set of candidate paths to every viable destination. The

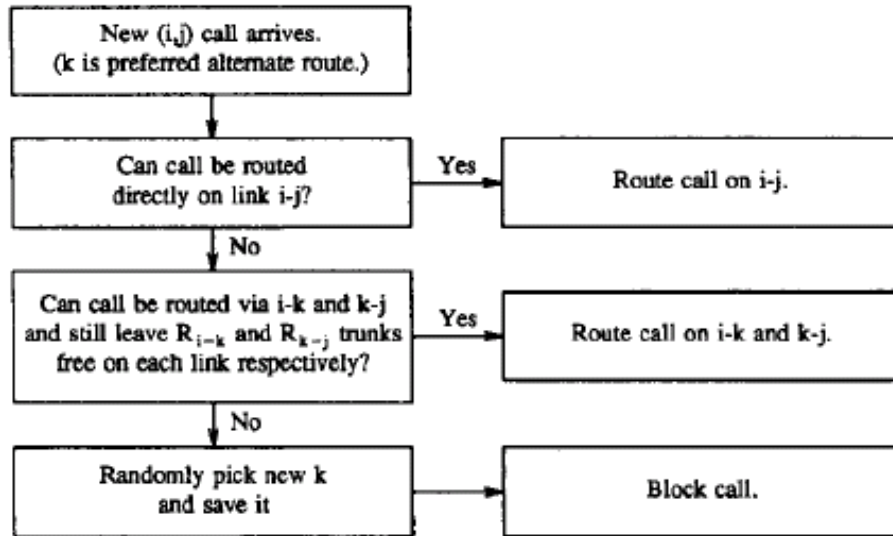


Figure 3.1 Dynamic alternative routing algorithm flow chart [76]

candidate path selection is a significant determinant in localised QoS routing whilst also having a noticeable effect on its performance. Further content regarding candidate path selection schemes can be consulted in [74, 75]. Several localised QoS routing schemes are discussed in the following sections.

### 3.3.1 Dynamic alternative routing

The initial concept of utilising localised information pertaining to routing has in fact been implemented in telephony supported networks [77, 78]. Telephony networks employ a scheme which transports a flow dependent on the feedback communicated from the preceding flows that received acceptance or rejection.

With the DAR [76] approach, the source node attempts to route a call via a straight unitary-link route to the destination. When the call isn't successfully given a route, then a two-link path is selected to route the call. Whereby if the call can't be given a route on the favoured two-link path, the call receives a block which results in another two-link path being chosen at random from all two-link paths and this becomes the new preferred two-link path.

The favoured path is used on every occasion to route a flow to its destination; the preferred path is always retained in the memory of the source node of every destination.

### 3.3.2 Learning automata based routing

According to the learning automata method [79], upon the arrival of a flow it is given a route on a path, denoted by  $r$ , dependent on a probability distribution denoted by  $p_r$ . The probability distribution is kept up to date by utilising feedback regarding whether the flow is accepted or rejected. The learning automata scheme employs a very simplified method of indicating a reward to a path if it is accepted, and indicating a penalty if the flow is rejected. The updating formula for path  $i$  chosen at time  $n$  if the flow accepted is:

$$p_i(n+1) = p_i(n) + a(1 - p_i(n)) \quad (3.1)$$

$$p_j(n+1) = (1 - a)p_j(n) \quad j \neq i \quad (3.2)$$

<pre> PROCEDURE PSR-ROUTE()   Select an eligible path <math>r = wrrps(R^{elg})</math>   Increment flow counter, <math>n_r = n_r + 1</math>   If failed to setup connection along <math>r</math>     Decrement failure counter, <math>f_r = f_r - 1</math>   If failures reached limit, <math>f_r == 0</math>     Remove <math>r</math> from eligible set, <math>R^{elg} = R^{elg} - r</math>   If eligible set is empty, <math>R^{elg} == \emptyset</math>     Reset eligible set, <math>R^{elg} = R</math>   For each path <math>r \in R</math>     Reset failure counter, <math>f_r = \gamma_r</math> END PROCEDURE </pre>	<pre> PROCEDURE PSR-PROPO-COMPU()   For each path <math>r \in R</math>     Compute blocking probability, <math>b_r = \frac{\eta \gamma_r}{n_r}</math>     Assign a proportion, <math>\alpha_r = \frac{n_r}{\sum_{\tilde{r} \in R} n_{\tilde{r}}}</math>   Set target blocking probability, <math>b^* = \min_{r \in R} b_r</math>   For each alternative path <math>r' \in R^{alt}</math>     If blocking probability <i>high</i>, <math>b_{r'} \geq b^*</math>       Decrement failure limit, <math>\gamma_{r'} = \gamma_{r'} - 1</math>     If blocking probability <i>low</i>, <math>b_{r'} &lt; \psi b^*</math>       Increment failure limit, <math>\gamma_{r'} = \gamma_{r'} + 1</math> END PROCEDURE </pre>
--	---

(a) Proportional routing

(b) computation of proportions

Figure 3.2 PSR Pseudo-code [5].

If the flow is rejected the updating equation is:

$$p_i(n+1) = (1 - \varepsilon)p_j(n) \quad (3.3)$$

$$p_j(n+1) = \frac{\varepsilon}{r-1} + (1 - \varepsilon)p_j(n) \quad j \neq i \quad (3.4)$$

### 3.3.3 Proportional Sticky Routing

Nelakuditi et al presented the first localised algorithm involving QoS, known as the Proportional Sticky Routing algorithm (PSR) [5]. The main principle of the PSR mechanism suggests that the route-level statistics, i.e. number of flows blocked, is the only QoS state data obtained by the source. The existing statistics are the basis for the algorithm to proportionally allocate the load from a source to

a destination amongst numerous paths according to their flow blocking probability. Therefore, the PSR algorithm needs every node to sustain a predefined set of candidate paths  $R$  to each destination. Supposedly, routing algorithms structured to choose a short path show better performance over the algorithms that are not burdened with additional path length [22, 80]. Two types of paths are differentiated by PSR. These are minhop paths  $R^{min}$  (the shorter paths) and alternative paths (paths with longer length)  $R^{alt}$ , where  $R = R^{min} \cup R^{alt}$ . Thus, the algorithm selects minhop paths and consequently decreases the so-called ‘knock-on’ cascade effect rising from alternative paths utilisation by some sources. It involves other sources, where minhop paths share links with alternative paths, and prompts them to use these alternative paths as their minhop paths [5, 81].

The PSR algorithm encompasses two steps: proportional flow routing and calculation of flow proportions. Proportional flow routing is a cycled process, where each cycle is of a certain length and presented by a selection of the set of eligible paths  $R^{elig}$ , followed by directing the flows along the preferred paths. A prearranged proportion  $\alpha_r$  serves to choose a path  $r$  with programmed frequency.

At the start, all the candidate paths are eligible and assigned a maximum permissible flow blocking parameter  $\gamma_r$ . It verifies the permitted number of blocked flows directed along the path before it is converted into an ineligible path.

For each minhop path,  $\gamma_r$  is set to  $y$ , a configurable system parameter. For each alternative path, the value of  $\gamma_r$  is dynamically set between 1 and  $y$ . A new cycle is reset with  $R^{elg} = R$ , and when the old one is finished  $R^{elg}$  becomes empty. An observation period includes the number of cycles  $n$ , and is terminated by a new flow proportion for each path  $r \in R$ ,  $\alpha_r$ , which is calculated based on its observed blocking probability  $b_r$ .

After the completion of each observation period, the PSR mechanism adjusts the minimum hop paths flow proportions to equalise them with their blocking probability ( $\alpha_r b_r$ ). For alternative paths, flow proportions are regulated by the minimum blocking probability among the minimum hop paths,  $b^*$ , i. e. for each  $\gamma \in R^{alt}$ , if  $b_r < \Psi b^*$ ,  $\gamma_r = \min(\gamma_r + 1, y)$ . If  $b_r > b^*$ ,  $\gamma_r = \max(\gamma_r - 1, 1)$ , where  $\Psi$  is a configurable factor to minimise the ‘knock-on’ phenomenon under system overloads.

Available sources dispute the benefit of the PSR algorithm [5]. Even though the application of PSR improves routing, this particular technique employs Erlang’s loss equation for calculating flow proportions, based on the steady-state of the blocking probability distribution [4] which can lead to certain complications in the case of non-Poisson or bursty traffic. Besides, as the number of the candidate paths decrements with each cycle, it becomes necessary to normalise flow

proportions calculated at the beginning of the cycle as soon as a path becomes ineligible. Due to the fact that flow distribution may not recall the prearranged proportion, the path which last happens to be ineligible is inclined to receive a higher number of flows than the others.

### 3.3.4 Localised Credit Based Routing

The credit based routing (CBR) [4] 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 based 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 the following condition is not satisfied.

$$P^{\min}.credits \geq \Phi \times P^{alt}.credits, \text{ where } \Phi \leq 1 \quad (3.5)$$

The symbol  $\Phi$  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 as  $0 \leq credits \leq MAX\_CREDITS$ .

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.

The main problem with CBR is that a path's credits are only updated each time that path is selected. If a path is selected infrequently then its credit value will become stale leading to errors in the selection process.

### 3.4 Summary

In this chapter we first mentioned the global QoS routing and presented a brief explanation of its algorithms. Subsequently, we came down to the description of the localised QoS routing approach as the second type of contemporary network routing. The method of the state information collection can be either global, which is based on global link state updates among routers in the whole network, or local, based on local path state recordings. WSP, SWP, DISP and QOSPF are examples of global QoS routing algorithms that select a feasible path relying on bottleneck bandwidth.

Localised QoS routing methods, such as the learning automata method, PSR and CBR deduce the state of the network and compute path selection by local information that is collected by source nodes. CBR and PSR use the blocking probability as a factor in selecting the routing paths. Having up-to-date information would not be possible with CBR or PSR algorithms working with either credit or flow proportion respectively, therefore our proposed algorithms presented in chapter 5 use residual bandwidth as the direct QoS criterion to select routing paths.

# **Chapter 4**

## **Modelling and Simulation Environment**

### **4.1 Introduction**

This chapter discusses a simulation model of QoS routing which will allow us to measure the functioning of the proposed algorithms. In addition, this simulation will allow us to contrast the suggested algorithms with other current algorithms subjected to varying configurations. On the underlying principle of this model considerable simulation experiments are conducted by employing various forms of network topologies and traffic distributions, while subjected to an extensive range of traffic loads. The simulation model comprises three parts: The simulator package forms the first part, which includes the implementation of global and localised QoS routing algorithms and this can be utilised to measure their performance in a practical way. The second part relates to the simulation performance metrics which are conveyed as evaluation metrics which can be gathered and presented by the simulator application. The final part, relating to the simulation model, contains simulation arrangements and models that can be

utilised in other research to measure how new localised QoS algorithms perform.

The subsequent sections will discuss the simulation model in more depth.

## **4.2 Simulator design**

The initial objective in producing a simulator is to facilitate the analysis of localised methods by employing networks containing a medium to large quantity of network nodes.

Furthermore, the aim is to represent a model containing large numbers of flows (in excess of 2 million) to give reliable computations, and to viably represent current networks which are characterised by large capacities and can maintain a substantial number of flows. By taking these aims into consideration, it was realised that the current simulation packages afforded to us (NS-2 simulator [82] and other network simulators such as OPNET [83, 84]) would necessitate a great deal of customised specifications and the inclusion of missing elements. In addition, given their focus on low-level modelling (i.e. packet-level specifications), these simulation packages are constrained by scalability issues when analysing extensive networks containing large numbers of simultaneous flows. Producing a new simulator necessitates substantial energy, and the operation of testing the complete code is a time consuming function. Thus we chose to adopt a more mid level approach and selected a generic-type network

simulator which has the provision for the fundamental functions of modelling the network elements yet allows us to focus on producing the significant elements of our simulator. In realising this aim, we produced our simulation algorithms using OMNeT++ [85-87].

### **4.3 Simulation Environment**

OMNeT++ (Objective Modular Network) is a component-based, modular discrete event network simulator with an embeddable simulation kernel and Graphical User Interface functionality. An OMNeT++ simulation is developed on C++ foundations and out of hierarchically nested modules. Modules are programmed in C++ and utilize messages as a method of communication amongst each other. A node retains an arbitrary level of gates which function to transport messages via links to other nodes. A network topology that holds gates, links and modules, is detailed in the Network Description (NED) language. OMNeT++ is a discrete event-powered simulator providing a great array of functionality and features for the simulation of components of a communication network, nodes, links, and data packets.

## 4.4 Simulation Structure

The conceptual model pertaining to the simulator is illustrated in Figure 4.1. It shows the significant modules and provides a general summary of the simulation procedure, where each module executes a distinctive task as discussed below.

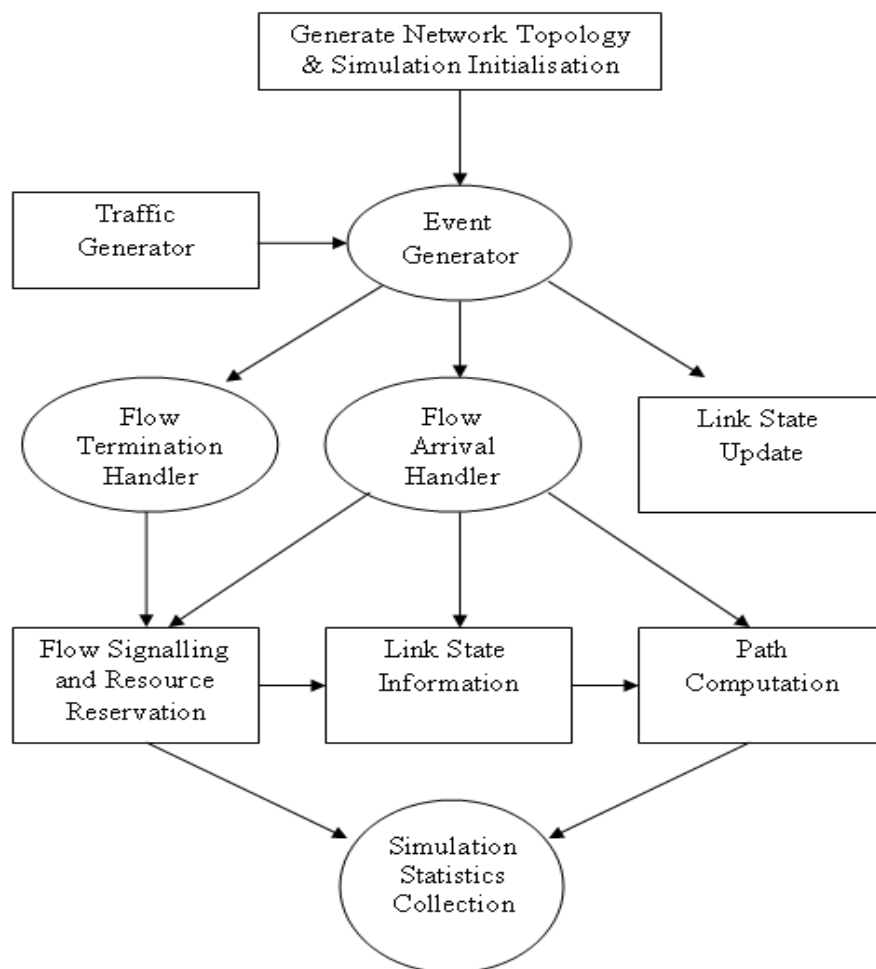


Figure 4.1 Functional components of simulator

### **Simulation initialization and network setup**

This module carries out the preliminary processes for the simulation. In this block the variables used in the entire simulation process are initialized. It begins by producing an unsystematic topology or interpreting the created topology using the assigned NED file. The Doar-Leslie model [88] is employed to generate the random topology in the following manner; initially, the requisite number of nodes are positioned randomly on a plane, and nodes are placed a specified distance ( $d$ ) apart. The Euclidean distance  $d(x, y)$  among nodes  $x$  and  $y$  and the maximum distance among any two nodes ( $F$ ) across a network are also calculated. Following this, the probability of including a link  $(x,y)$  is determined conforming to the probability  $P = \beta(k_e / |N|) \exp(-d(x,y)/\alpha F)$ . Coupling  $\beta$  values above one, which the Waxman idea did not support [89], with very small  $\alpha$  values lead to producing practical graphs [90]. This equation or strategy applied to every pair of nodes, which are distributed randomly over an x-y coordinate grid, to calculate the Euclidean distance ( $d$ ) between the nodes. The required average node degree ( $e$ ) must be assigned by the user. Using the selected values of  $\alpha$ ,  $\beta$  and  $e$ , we varied the  $k$  value and ran our simulation to get an average node degree close to  $e$  to generate the required, practical graphs. Each graph generated was tested to guarantee that each node is connected to at least one other node. The graphs generated have a reasonable distribution of short and long links.

The first values are ascribed to the topology, such as link capacities, link delays and other simulation variables. Routing tables of all nodes are constructed throughout these processes. Furthermore, the arrays of candidate paths for every source/destination set are generated on the parameterized topological data. This set is solely utilised by localised algorithms.

### **Traffic Generator module**

This module computes the characteristics of each arriving flow on the basis of the stated traffic variables. The traffic generator module maintains the features, as follows:

- Arrival process: In this process the traffic can be simulated either as a Poisson or bursty traffic.
- Stream flow duration: streams can have either an exponential or Pareto duration distribution.
- Bandwidth requirements: either uniformly distributed or fixed.
- Selection of Source/destination: is selected in a random manner.

### **Event Generation**

The three principal activities of the simulator are:

- Flow arrival: The arrival of a flow causes the invoking of this handler which relays this flow with its bandwidth requirements,  $b$ , to the path computations module to compute the optimum feasible routes. If the path computations module generates a path regarded to be viable, the handler will activate the flow signalling and resource reservation module to indicate the flow.
- Flow termination: Invokes the flow indicating and resource reservation module to deliver resources held back for the terminating flow.
- Link-state update event: Invokes the Link-state updater module providing the route computation module uses an algorithm that requires global state information, i.e. the WSP algorithm. Generally, maintaining the global link state information is performed with flooding or spanning tree strategies executed by a certain routing protocol. Nevertheless, this influenced the scalable feature of the simulator because of the extensive level of intricacies and message relaying that is required to be simulated in low-level detail. Given that we are not concerned with actions of such particular strategies, the link-state updater module does not deploy any specified strategy, but rather every link is related to a variable indicating its broadcasted accessible bandwidth. This variable is maintained at regular intervals dependent on the update interval to express the available

bandwidth. The path computation module utilises this variable to carry out the task of choosing the path. Thus we can represent via simulation the effect of stale global state without having any effect on the scalability of the simulator.

### **Path computation module**

This stage of the simulation implements several localised QoS routing algorithms and one global QoS routing algorithm, which is the WSP algorithm. The illustrated localised algorithms employ the information gathered by the localised state collection module while WSP utilises the global state aided by the link-state updater module.

### **Flow signalling and resource reservation module**

The functions of this module are for resource reservation, admission control and signalling policy. It is invoked once the path computation model locates a viable route which fulfils the QoS requirements. The signalling procedure begins hop-by-hop, starting at the source node, to reserve network resources for the connection arrival.

### **Bandwidth Metric**

We make the assumptions that the desired bandwidth is  $b$  and every link  $\ell$  across the network contains an available bandwidth  $\text{bw}(\ell)$ . When the signalling message moves through the chosen path  $p$ , every node performs an admission inspection on the outgoing link to ensure it has the required bandwidth. If accessible bandwidth (outgoing link) equates to or is more than the required bandwidth, the node holds back the bandwidth ( $b$ ) for the subsequent newer flow so that  $\text{bw}(\ell) = \text{bw}(\ell) - b$  and the message is transported to the subsequent node in the route. This module allows the flow given that the links on the chosen route (path  $p$ ) have sufficient bandwidth and, if not, a failure message is relayed to the source node which then releases the reserved bandwidth so that  $\text{bw}(\ell) = \text{bw}(\ell) + b$  and the flow is refused.

### **Delay Metric**

We hold the assumption that the delay constraint ( $\text{QoS\_Delay}$ ) and every link ( $\ell$ ) throughout the network exhibits a delay  $d(\ell)$ . When a signalling message moves across the chosen path  $p$ , every node performs an admission analysis over the outgoing link adding its latency (queuing delay at the node plus the propagation delay over the link) to the preceding total latency to ensure the flow does not get delayed in excess of the requested delay parameters. If the delay over the

outgoing link is less than or equal to the requested delay constraint the message is passed to the next node in the path. This module allows the flow, given the delay across the chosen path  $p$  is not greater than the desired delay condition, so that  $\sum_{i \in n} delay(i) \leq QoS\_Delay$ , with  $n$  indicating the number of links across the chosen route, and also the delay constraint of existing flows is not surpassed, or else a rejection message is relayed to the source node and releases the reserved resources and the flow is refused.

It should be made clear that we are here talking about the mean delay and not the instantaneous delay since the later would need to be represented by a probability distribution and so change from instant to instant. This would make it meaningless as a QoS metric.

This approach also engages with the flow termination module when the flow time period of a flow has expired to make available resources held back in reserve by that flow. Attention should be paid to the fact that this module is not responsible for re-routing the flow to another possible path once an unsuccessful setup message appears and as a consequence the flow is not accepted. Despite rerouting flows lowering the probability of blocking, it would also lead to higher signalling overhead.

### **Localised state collection**

Localised state collection, as indicated by the name, provides support for localised routing algorithms and is tasked with gathering local information for the localised routing algorithms. It should be considered that this module engages with the flow indicating and resource reservation module in order to gather information regarding whether a flow is accepted or rejected.

The local state information can also involve staleness due to lack of updates. For example, the blocking probability of a specific candidate path or the flow proportion measure is only updated every time the path is used. In the meantime the traffic levels on paths that share links might have changed significantly and this would impact on the QoS of the specific candidate path.

Having up-to-date information would not be possible with CBR or PSR algorithms working with either credit or flow proportion respectively, unlike our proposed algorithms, presented in the next chapter ,which use residual bandwidth in the path selection and so work with the QoS metric directly rather than indirectly.

### **Simulation statistics collection module**

This model is responsible for gathering appropriate and applicable computed data from the simulation, including blocking probability and average transported

traffic. It is also responsible for producing these metrics as a function of time. This ensures, for instance, to map the transported traffic across the time axis throughout the simulation or the use of a specified link as a result of quick changes in the offered traffic.

## 4.5 Graph Model

The simulation produced is considered to model the framework of the actual Internet subjected to varying factors. 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. Recently, interest has arisen in the simulation of a more practical topology in order to model the topology of the Internet. This has come about because the performance of routing algorithms could possibly generate inaccurate conclusions if the evaluation is performed on topology that is not applicable or suitable. Furthermore, it is a hard task to simulate the framework of the Internet given its rapid evolution [91-93]. Networks can be distinguished depending on topological characteristics. Therefore, node degree, clustering of nodes and shortest-path length between any two nodes are the main factors to summarise the characteristics of any network topology [94]. Current models can be deployed to simulate routing algorithms requiring a few or all the

characteristics mentioned. We will discuss the models that are appropriate and which are implemented in this study.

### 4.5.1 Random Topology

Several studies have used the same methods which have been conducted in [95, 96] to produce practical and logical networks based on random graphs. The random graph system is produced by introducing links with probability of a function of the Euclidean distance between any pair of nodes in the graph. By varying the distributions of probability on network graphs the implemented network graph models will also differ. The prominent random graph generators will now be described, which are the Waxman model [89], and the Doar-Leslie [88] model.

#### Waxman Graph Model

The random network topology generator is a model regarding the growth of computer networks. In this model nodes are placed randomly on a plane and the links between nodes are created based on probability depending on the node's Euclidean distance. The following equation gives the probability to generate a link between nodes  $x$  and  $y$ , as given by [89]:

$$P = \beta e^{-d(x,y) / \alpha F} \quad (4.1)$$

Here  $0 < \beta$ ,  $a \leq 1$ ,  $d$  is the distance from  $x$  to  $y$ , and  $F$  is the maximum distance between any two nodes in the graph. An increase in the parameter  $\beta$  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. Despite the fact that Waxman's model is extensively implemented due to its ease of implementation, it has, nevertheless, a fundamental constraint as the quantity of nodes increase, which can result in unrealistic node degrees.

### **Doar Leslie graph Model**

Doar and Leslie suggested a modification to the Waxman random model by adding a scaling factor so as to restrict the rise in the average node degree [88]:

$$P = \beta(ke / N) \exp(-d(x, y) / aF) \quad (4.2)$$

where  $k$  denotes the scale factor and  $e$  represents the average node degree.

### **4.5.2 Regular Topology**

A regular graph is a graph with each node having an equal number of neighbours. Therefore, a regular graph with nodes of degree  $e$  is called an  $e$ -regular graph or regular graph of node degree  $e$ . While a regular graph topology is not prevalent in the Internet, it is commonly employed to determine some characteristics of the Internet's capability. Some forms of regular graphs that are utilised in the

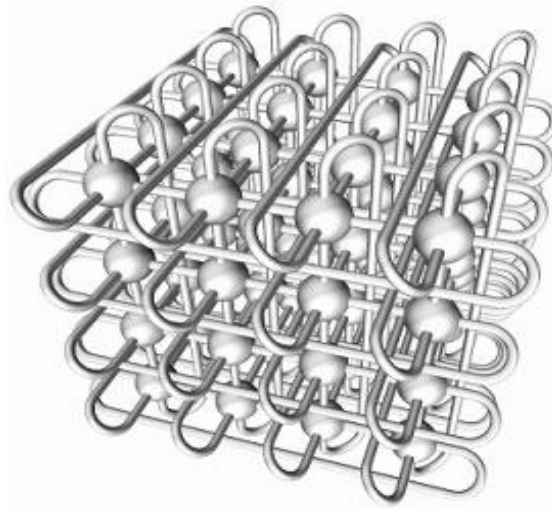


Figure 4.2 a 4-node Torus graph

simulation and testing of routing algorithms include Torus, Star, and Ring. Figure 4.2 illustrates a 4-node Torus graph.

The Torus topology has been studied widely to evaluate the performance of QoS routing as it gives a variety of path lengths for many source-destination pairs. A  $7 \times 7$  node Torus topology is used in our simulation.

### 4.5.3 ISP Topology

Most Internet Service Providers' (ISP) topologies are designed to optimize the performance of network traffic, guarantee security and ensure reliability. This topology is a well known network model and depicts a single autonomous system domain for ISP networks in the U.S. Within each ISP domain there can be hundreds of interconnected routers and points of presence [97, 98]. This type of

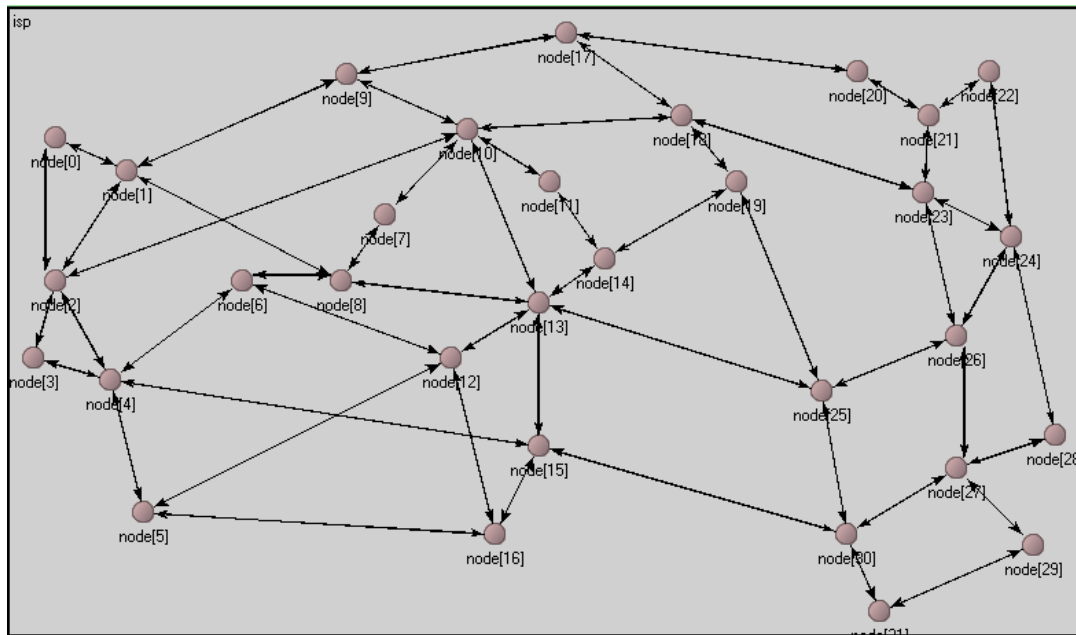


Figure 4.3 the ISP topology

ISP topology has been widely employed for simulating routing algorithms [99, 100]. This graph model will be used in our simulation to evaluate the performance of the proposed localised QoS algorithms. The ISP topological model is illustrated by Figure 4.3.

## 4.6 Performance Metrics

In the QoS routing system, a route only receives acceptance on the explicit condition that it fully meets the requisite QoS. If it fails to satisfy the QoS conditions then the path is not viable and thus is not used and the flow is rejected. In the second scenario, mentioned above, important network resources have been

consumed in the process of determining the path; this is an unnecessary and costly process. The suggested algorithms can be evaluated by using flow blocking. A flow will be rejected in the situations where one of the links (along a route) from source to its destination point fails to meet the requested bandwidth or when the chosen route exceeds the requested delay conditions. A clear method of measuring such overhead costs is from the ratio of the number of flows blocked  $B$  and the total number of flows that arrived at the network  $T$  as measured between specified time duration, as given by (4.3). This ratio conveys a reasonable estimate of the flow blocking probability and also serves as a calculation of the efficiency of the QoS algorithm that has been used.

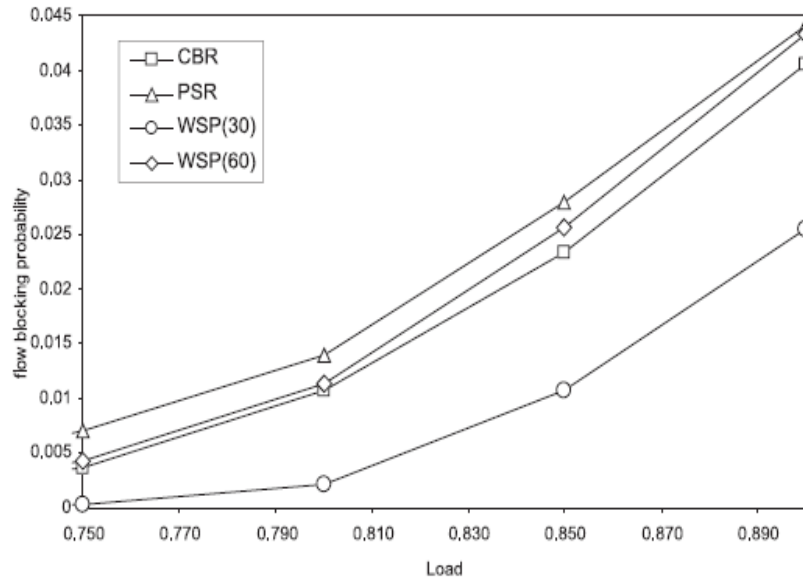
$$\text{Flow blocking probability} = \frac{|B|}{|T|} \quad (4.3)$$

## 4.7 Simulator Validation

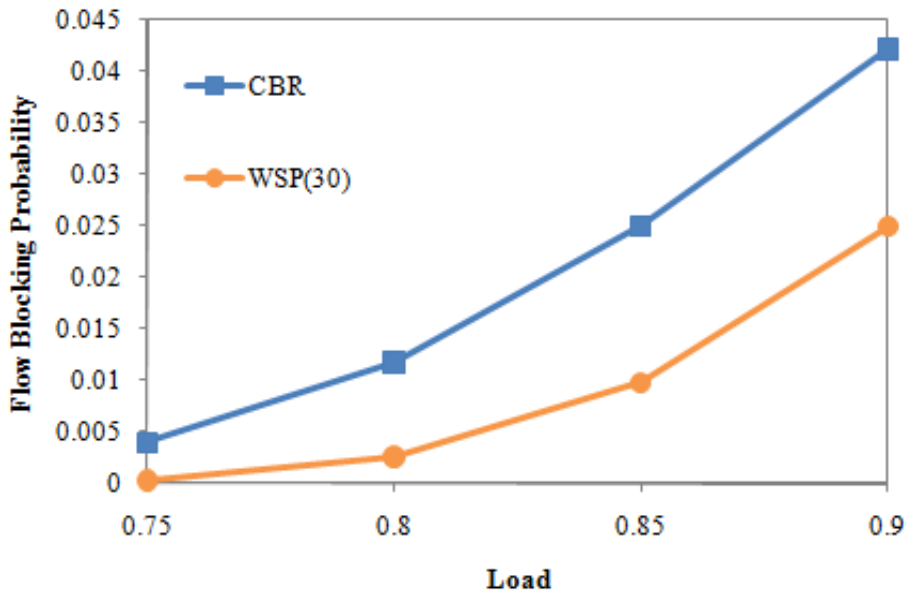
The validation function serves to provide an assurance that the simulation operates and functions according the requirements, via the assurance that there exists no major variance among the results from the simulation model and results for which there is a high confidence of their validity. Given that the simulator is constructed on top of the OMNeT++ simulator, we thus took the assumption that the fundamental simulation engine and the basic functions delivered by OMNeT++ are inherently accurate. Our assumption is deemed fair given that

OMNeT++ has been widely adopted in the academic research sphere for a significant time and also because many research scientists have undertaken research projects by utilising this specific simulator. For further details relating to this, the OMNeT++ web portal can be consulted [85]. For the purpose of this particular thesis, the validation test process has been executed by performing simulations using OMNET++ in order to determine that the simulator gives correct results for the localised and global QoS routing algorithms. In order to determine the validation using the credit based routing algorithm (CBR), the results gathered have been considered in comparison to the results reported in [4]. Likewise, for the case of the Widest Shortest Path algorithm, by adopting the identical simulation parameters and arrangements as conveyed in [4] by the creators of the CBR algorithm. We replicated the simulations by employing our simulator and we conclude that the results correspond closely to those obtained independently in [4], as detailed in Figure 4.4.

The results on Figure 4.4(b) were obtained from a simulation of 2,000,000 flow arrivals with the first 200,000 flow arrivals used as a run-up period to stabilise the algorithms. The 95% confidence intervals were too small to be visible with the scales used and so do not appear. The above applies to all subsequent simulation results presented in the thesis.



(a) Original results in [4].



(b) Verified results.

Figure 4.4 Simulator validation results.

## 4.8 Summary

To research communication behaviour, two principal methods exist, namely analytical modelling and the simulation method. The latter generally receives preference given that it has the capacity to evaluate more intricate and difficult systems that sometimes prove difficult to manage using analytical methods without using assumptions that cannot be held as being pragmatic. The process of simulating the performance of QoS routing algorithms comprises two processes. To begin with, QoS routing algorithms have to be simulated and these involve certain assumptions and specified parameters. For the task which is required to simulate routing algorithms, then a network model has to be utilised. There are several graph models implemented for the modelling of a computer network. On the basis of this information, OMNet++, an open-source communication network simulation environment, has been utilised with the aid of the C++ language in order to perform the simulation and analysis of the delivery of the suggested algorithms.

This chapter has discussed the simulator design including the way in which it was validated. Several forms of network topologies, parameter conditions and the blocking probability as a measurement metric employed in the performance evaluation have also been discussed.

# **Chapter 5**

## **New Localised algorithms based on Bandwidth as the QoS metric**

### **5.1 Introduction**

Periodic exchanging of global QoS state information and keeping it in a database at network routers is one of the problems in the global state routing. Several routing schemes [3, 9, 22, 53] that have been introduced for QoS routing need this exchange of the link state information among the routers in the whole network. This difficulty leads to flapping of routers and high communication overheads. Localised QoS routing is considered an alternative method to global state routing to reduce the problems mentioned above. Flow blocking statistics that are collected locally and provided to the network QoS state by the source nodes minimize the communication overhead and eliminate the need for the routers to update and keep a database of QoS state. A considerable amount of literature has been published on QoS routing [2, 7]. The basic concepts of processing local information have been used in many different mechanisms of telephone technology, which was applied long before its implementation in computer

networks [81]. Both PSR [5] and CBR [4] represented the idea of localised routing in computer networks and they are considered as relevant work to our algorithms. The CBR is the most relevant work to our algorithms. Its superior performance compared to PSR has already been shown in available sources [4, 5, 71]. For this reason we will use CBR as a standard to compare our mechanisms. Although the CBR algorithm, based on highest path credit, was designed to select one of the candidate paths, it still does not reflect the quality of the path in terms of the bandwidth or delay, both of which are examples of QoS constraints and mirror the path quality directly. In addition, it does not seem a logical approach to use indirect representation of path quality by the use of path credits.

This chapter presents new localised routing mechanisms and the following section illustrates the concept of the algorithms and identifies the key determinants of these algorithms to choose the optimum path from the candidate path set.

## **5.2 Proposed algorithms**

The new algorithms are source routing algorithms, where the source node takes the routing decision. The internet application of these algorithms can possibly be mediated by one of the various traffic engineering techniques such as Multiple-Protocol Label Switching (MPLS) [101], which is an example of a protocol used to set up one or multiple paths between each pair of source and destination nodes.

CBR and PSR use the statistic of blocking probability as a factor in selecting the routing paths. Apart from their principles of function, in this chapter our proposed algorithms use residual bandwidth as the direct QoS guideline to select routing paths.

Also, an important aspect of our algorithms is that by using the metric of interest, such as bandwidth, any changes in this metric due to flows on candidate paths of other nodes can be conveyed back to the source node of all other candidate paths that may share part of their routes with the path of interest. In other words, any changes in bandwidth at any node  $x$  which is included in a candidate path  $P$  is relayed to all source nodes which include node  $x$  in one or more of their candidate paths. In this way all nodes always have up-to-date information concerning their candidate path set and this is a crucial point. Having up-to-date information would not be possible with CBR or PSR algorithms working with either credit or flow proportion respectively. CBR or PSR would only update the information related to their candidate path sets each time a path from the given sets is selected. This means the information could become out-of-date for paths that are only selected infrequently.

### **5.2.1 Highest Minimum Bandwidth (HMB) algorithm**

The HMB algorithm selects the highest minimum residual bandwidth among the candidate path set. The mechanism scenario starts when a setup message travels from source to destination along the outgoing links in the path. All the outgoing links in the single path are compared, to locate the link with the minimum residual bandwidth. Each selected link refers to a path in the candidate path set.

The HMB 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. This process starts at the source node by sending a set up message along the selected path. Each node in this path acts as a router to test if the following link has sufficient residual bandwidth for the flow. In the event that the residual bandwidth is not sufficient and does not satisfy the QoS requirements, a failure message is sent back to the source informing of the failure of this path. In the successful case, the bandwidth of the message will be reserved from the residual bandwidth and the message will be forwarded until it reaches its destination. The pseudo code for this algorithm is shown in Figure 5.1.

```

Initialize
  Set SelectedPath = P0
HMB ( )
1.   MinResidualBW = min (L.ResidualBW, MinResidualBW),  $\forall L \in P$  and
 $P \in R$ 
2.   if (SelectedPath.MinResidualBW < P.MinResidualBW)
3.     Set SelectedPath = P
4.     Route flow along SelectedPath

```

Figure 5.1 The pseudo code for the HMB algorithm.

The HMB algorithm selects the best path by selecting the link that has the largest residual bandwidth among the selected links with the minimum residual bandwidth. In this algorithm a predefined set of candidate paths  $R$  are required by each source and destination pair. A variable  $P.MinResidualBW$  stores the minimum residual bandwidth link in each path  $P$  (line 1). Every path  $P$  is associated with this variable. First, path ( $P_0$ ) in the candidate path set is assumed to have the minimum bandwidth link. The comparison (in line 2) selects the highest minimum bandwidth between all the selected links in the candidate paths. The flow is routed along the selected path (line 4). Figure 5.2 shows the flow chart of the HMB algorithm. Importantly, preference has to be given to the link with the minimum residual bandwidth that satisfies the QoS among the selected paths. It ensures that the selected path can encompass the flow. CBR routes the flow based on a crediting scheme that rewards a successful path and penalizes the failed. Furthermore, CBR credits the whole path as one block, whereas our

presented algorithm chooses the path on the basis of the residual bandwidth in each link from the candidate path set.

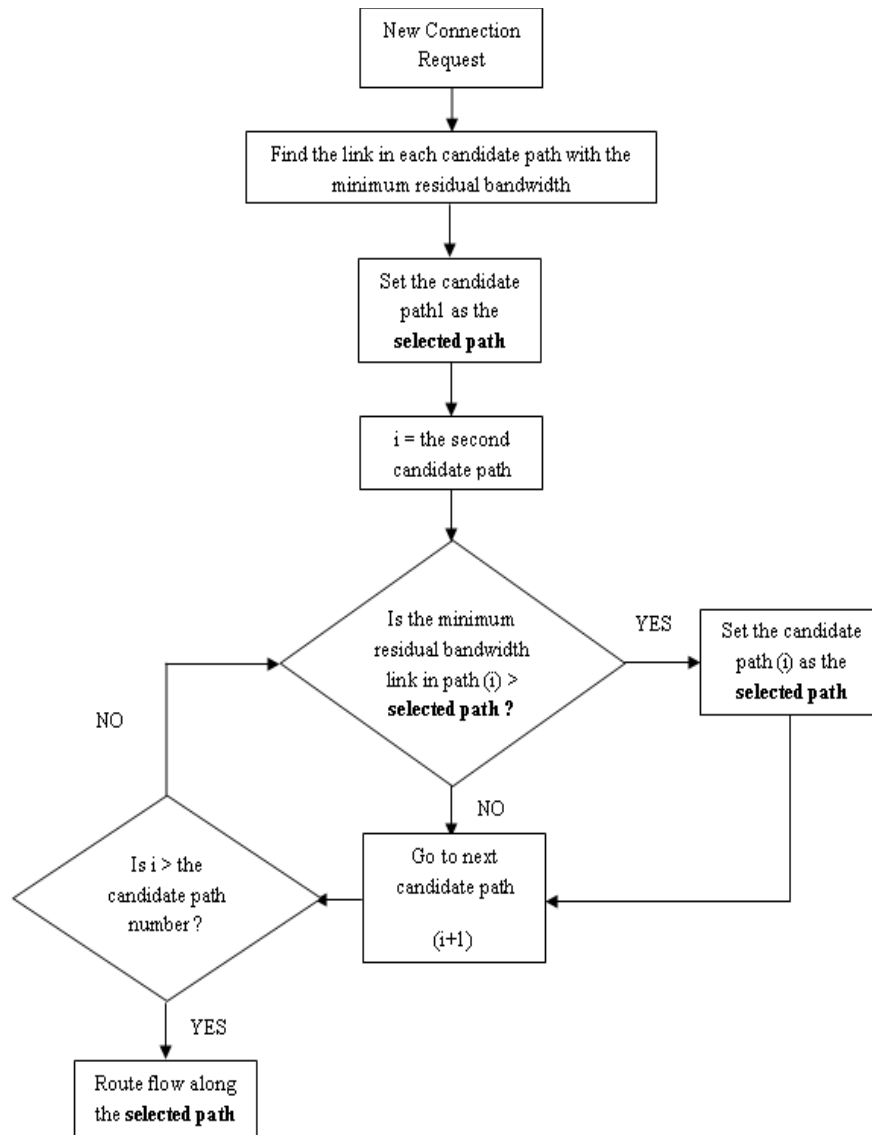


Figure 5.2 The flow chart of the HMB algorithm.

## 5.2.2 Highest Average Bottleneck Bandwidth History (HABBH) algorithm

In HABBH, each link in the path is given an interval (sliding window) in order to compute the average residual bandwidth for that particular link. The best choice path is assigned on the basis of the window results. The path which generates the highest average bottleneck residual bandwidth becomes the candidate path along which to direct the arriving flow. HABBH performs the screening of residual bandwidth across the network and constantly maintains each link's average residual bandwidth. By taking into account the average residual bandwidth for each link it indicates the viability of the path. The pseudo code for the HABBH algorithm is shown in Figure 5.3.

```

Initialize
  Set SelectedPath = P0
HABBH ( )
1.   L.ListBWHistory.Add(L.ResidualBW) , LP
2.   if (L.ListBWHistory.Count > HistoryWindowSize)
3.     Remove old value from the window
4.     P.MinExpectedBW = min (L.ListBWHistory.Average , P.MinExpectedBW)  $\forall L \in P$  and
P \in R
5.   if (Selected Path.MinExpectedBW < P.MinExpectedBW)
6.     SelectedPath = P
7.   return SelectedPath

```

Figure 5.3 The pseudo code for the HABBH algorithm.

A principle of the localised QoS routing mechanisms is that each pair of source and destination nodes needs a pre-established set of candidate paths  $R$ . In HABBH the average residual bandwidth link is the key determinant which is assigned to every path  $P$  in the candidate path set. A comparison is performed by the setup or the test message over the path links to obtain the lowest residual bandwidth link history for that path (line 4). These selected links are compared with each other in order to obtain the path that has the highest average bottleneck residual bandwidth (lines 5). The flow is routed along the selected path (line 7). Figure 5.4 shows the flow chart of the HABBH algorithm.

### 5.3 Performance evaluation

As mentioned previously, the superior performance of CBR compared to PSR has been shown in [4, 5, 71]. For this reason we used CBR as a standard localised algorithm to compare our mechanisms. Also, WSP [53] was deemed a relevant and well used contemporary global algorithm with which to benchmark our schemes in the simulation. This algorithm selects the least distance feasible path with the lowest hop count between paths that fulfill the bandwidth restrictions. Where there are several paths with an equal hop count then the path with the maximum available bandwidth is chosen. The widest path is selected when there are several paths with the same distance. In WSP ( $x$ ),  $x$  is the number of time units

that represents the interval that the WSP algorithm uses between updates of the link state information.

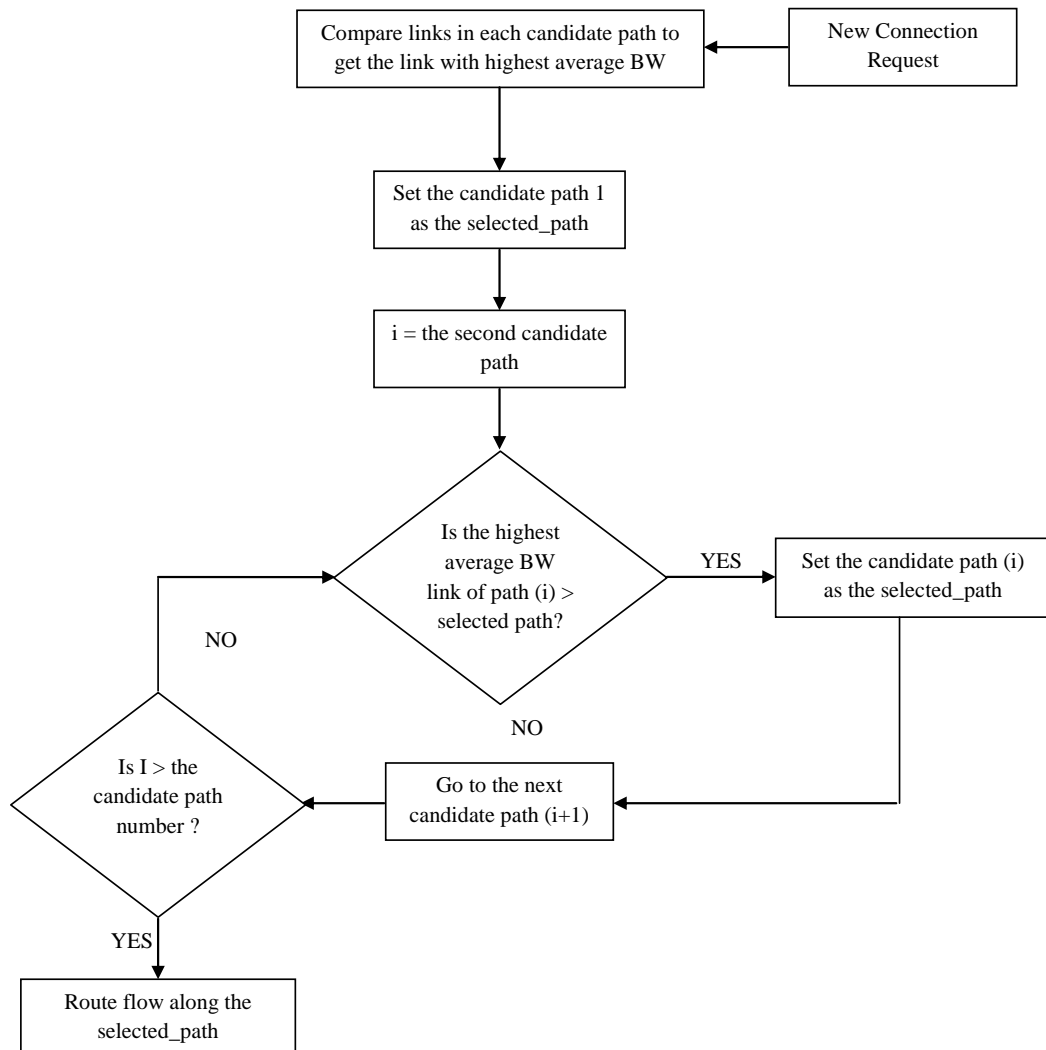


Figure 5.4 The flow chart of the HABBH algorithm

## 5.4 Methodology

In this section we describe the simulation characteristics and the network topologies used in our experiments.

### 5.4.1 Simulation environment

Our localised QoS routing algorithms were programmed to operate at flow level. Resource reservation, admission control, and selecting of the preferred path are based on the algorithms (HMB and HABBH), and simulated using the discrete-event simulator OMNeT++ [102].

### 5.4.2 Network topologies

The underlying network topology might affect the functionality of the routing algorithms. Therefore, we simulate different topologies of both regular and random networks and run our algorithms over these types of networks. In the selected random topologies the Doar-Leslie Model [88] plays the main part of identifying the probability of the node degree between any pair of source and destination nodes. The Waxman random graph scheme [89] has been modified by adding a scaling factor to produce the Doar-Leslie Model. The ISP topology is one of the regular backbone networks used in our study and has previously been used in various studies [9] [103]. Therefore, all networks were implemented in

<b>Topology</b>	<b>Nodes</b>	<b>Links</b>	<b>Node degree</b>	<b>Avg. path length</b>
ISP	32	108	3.375	3.177
Torus	49	196	4	3.5
Rand45	45	172	3.822	2.692
Rand80	80	482	6.025	2.99

Table 5.1 The characteristics of the topologies used.

OMNet++ combined with C++. Table 5.1 shows the characteristics of the topologies used.

### 5.4.3 Simulation Characteristics

There are some assumptions applied to our work to simplify the simulation. In all networks, all simulated links are assumed to be bidirectional and of the same capacity with  $C$  units of bandwidth in each direction ( $C=150$  Mbps). The flow bandwidths are distributed uniformly within a range  $[0.1, 2$  Mbps], whereas, the arrival rate of flows reaches each source node according to a Poisson process with rate  $\lambda$ . Nodes in the networks can be either sources or destinations. The flow service time is exponentially distributed with mean  $1/\mu$ . A random source node is chosen, and the destination node is selected randomly from the rest of the nodes.

Following [22, 104], the mean offered load of the network  $\rho = \lambda N \bar{b} h / \mu L C$ , where  $N$  is the number of nodes in the network,  $b$  is the mean bandwidth per flow,  $h$  is the average hop count per flow,  $L$  is the number of links in the network. Various ranges of loads are used in our simulation to estimate the new algorithm. These values of loads chosen are based on the network characteristics, where some networks need to apply a heavy load to evaluate the algorithm performance, and this explains the reason for choosing different load values in the represented networks. All paths between each source-destination pair having a maximum length, at most, of one hop more than the minimum number of hops, are chosen as the candidate paths to conform to previous studies [5]. In this thesis we also introduce some methods of candidate path selection to improve the CBR. The same methods are applied on our offered schemes. Selection of disjoint paths between each pair of source and destination nodes shows progress in reducing the blocking probability. Recalculation is the second method which aims to determine the amount of blocking in each path between each pair of source and destination nodes in the network, and subsequently modify the set of candidate paths by replacing the path of highest blocking by a lower blocking path.

### 5.4.4 Performance Metrics

The performance of the QoS routing algorithms is estimated by determining the overall flow blocking probability and bandwidth rejection probability. The flow blocking probability is calculated as the ratio of the number of flows blocked,  $B$ , to the total number of flows that arrived at the network,  $T$ , as in (5.1) and (5.2).

$$\text{Flow blocking probability} = \frac{|B|}{|T|} \quad (5.1)$$

$$\text{Bandwidth rejection probability} = \frac{\sum_{i \in SB} \text{bandwidth}(i)}{\sum_{i \in SR} \text{bandwidth}(i)} \quad (5.2)$$

Here,  $SB$  is the set of blocked paths and  $SR$  is the set of total requested paths, and  $\text{bandwidth}(i)$  is the requested bandwidth for path  $i$ .

### 5.4.5 Path selection methods

We introduce three methods of candidate path selection with the aim of supplementing our offered mechanisms and see if they offer any performance gains.

#### Selection of disjoint paths

Basically, each candidate path set has a certain number of paths connecting each pair of source and destination nodes. Common links which are shared by two or

more candidate paths are counted in this method. The number of common links is associated with each prospective candidate path and the paths with fewer common links are selected.

### **Recalculation of the set of candidate paths**

This method modifies the set of candidate paths by replacing the path of higher blocking by the lower blocking ones after calculating the amount of blocking in all possible paths between each pair of source and destination nodes in the network during a predefined interval.

### **Dynamic path selection method**

Basically, the candidate path set has a certain number of candidate paths between sources and destinations. In our dynamic method the amount of blocking in every path, between each pair of source and destination nodes is calculated. It is followed by modification of the candidate path set by replacing the paths of higher blocking with lower blocking ones. The main difference between the recalculation method described earlier and the suggested dynamic method is that the second technique replaces the set frequently on the basis of the given system variable value, the *RecalculatePath*. It gives the period or the number of connections requested to start the dynamic method, whereas recalculation changes the set only once during a predefined interval.

## 5.4.6 Simulation Results

### Impact of path selection methods

As can be seen in Figure 5.5, the selection of disjoint paths and recalculation of the set of candidate paths decreased the blocking probability in our new algorithms and also that of the existing CBR algorithm. Although the use of disjoint paths and recalculation of the set of candidate paths dramatically decreased the flow blocking probability for all network topologies, the best improvement of using these methods was observed in the Rand80 network shown in Figure 5.5. This is most likely because a random network would tend to give a better chance of selecting disjoint paths in the candidate path set than the more restrictive topologies of the ISP or the regular networks, such as the Torus. Moreover, the Rand80 network has the highest average node degree compared to other random networks used in our simulation model (Rand45) which thus gives Rand80 more options to benefit from the disjoint path method.

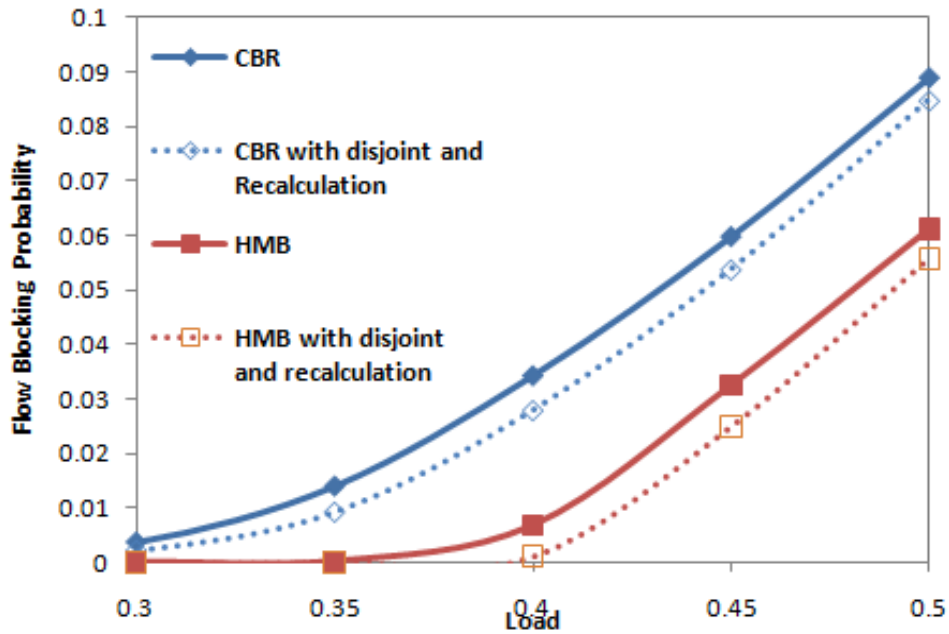


Figure 5.5 Impact of disjoint paths and recalculation in the candidate path set of different algorithms on Rand80 network.

Figure 5.5 also shows the impact of selection of disjoint paths and recalculation of the candidate path set on one of the presented algorithms (HMB) and the existing localised algorithm CBR. Joining both methods decreases the flow blocking probability in HMB and CBR algorithms. In other words, applying these methods in these localised algorithms leads to reduction in the flow blocking probability.

Although the recalculation method is applied, we keep the same number of candidate paths in the set. In our experiments, after the first 100,000 flows the recalculation method is applied to assist in reducing the blocking probability and

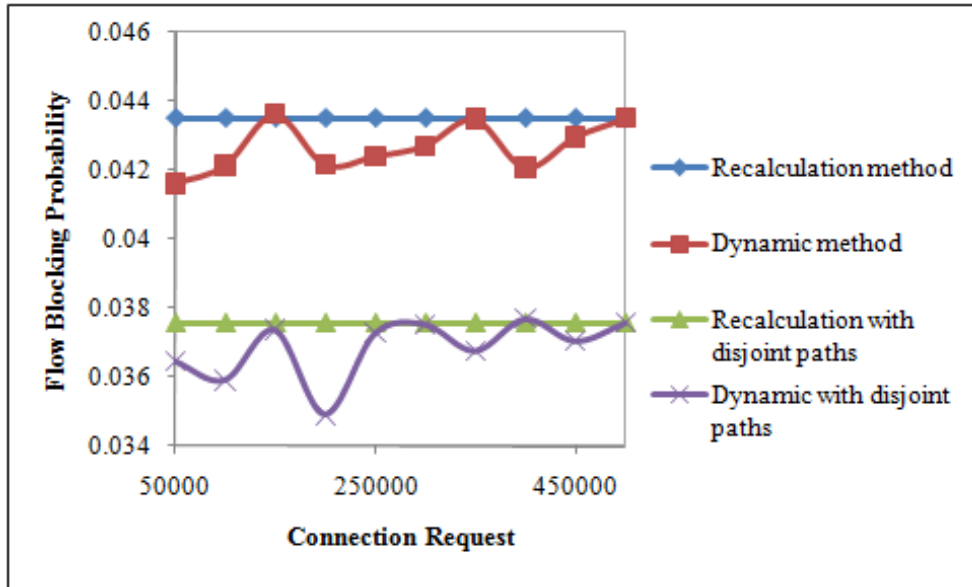


Figure 5.6 Dynamic method performance of HMB algorithm.

produces better performance. Each run simulates the arrival of 2,000,000 flows. After the 200,000 flows the simulation results are calculated, as mentioned above. Although use of the disjoint path technique improved the algorithm's performance, the same method combined with the dynamic approach gives superb results and reduces the blocking probability still further.

Figure 5.6 shows the benefit of the dynamic method and illustrates the performance of the HMB algorithm. We run a variety of experiments under fixed load 0.2 under the same network conditions. The recalculation method results stay the same, even if the system variable (number of connection requests)

increases. However, it shows improved performance in case of using the fixed candidate path set (Figure 5.5).

Although the dynamic method gives fluctuating results with every single change in the value of the dynamic system variable, it still performs better than a single recalculation of the path set.

The performance of the dynamic method fluctuates due to substitution in the candidate path set after a specific number of connection requests. When this substitution occurs, the performance improves. As the substituted candidate paths become more used the performance deteriorates again. Overall, the performance of the dynamic method is better than the recalculation method. In addition, combining the disjoint path with both recalculation and dynamic methods once again improves the disjoint path method.

Figure 5.7 shows the blocking performance of different algorithms using various ranges of update intervals in a Torus network for a fixed load of 0.9. WSP is the only global algorithm among the represented schemes and its blocking probability can be seen to increase with the increase in update interval. The performance of the other presented localised algorithms remains fixed and the update interval does not have an effect on them. Although WSP with small update intervals (lower than 60) performs better than CBR, our novel algorithm outperforms WSP,

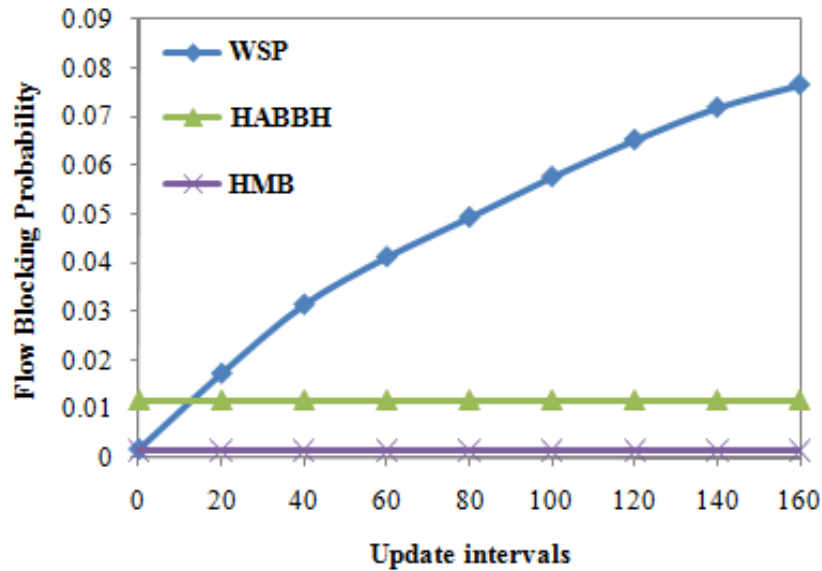
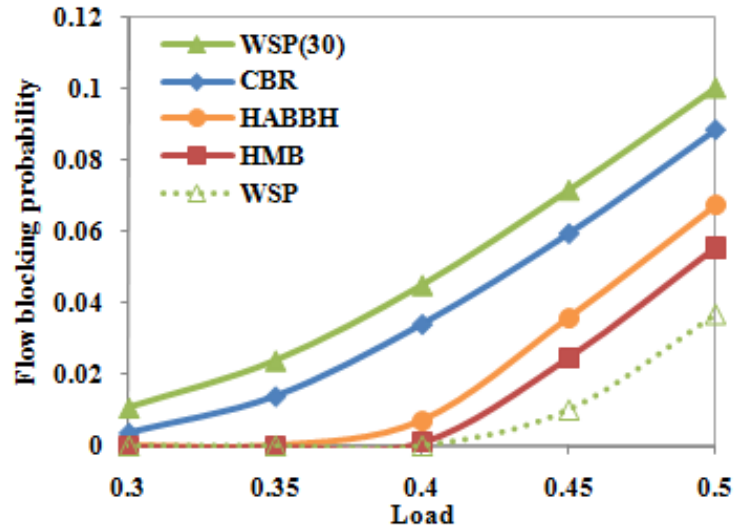


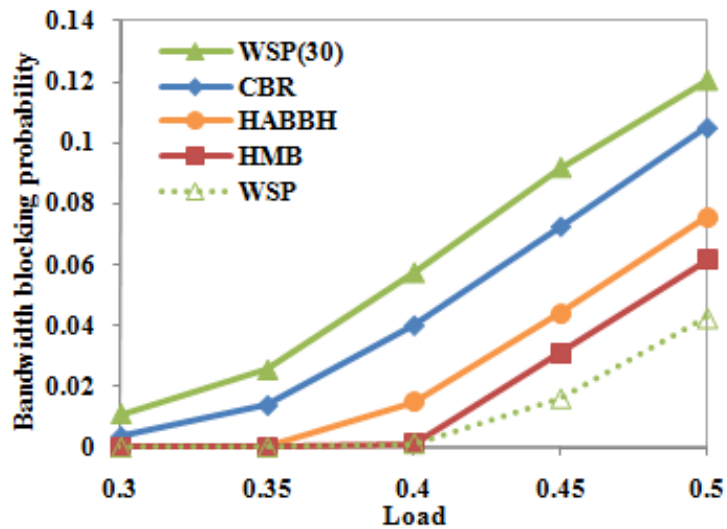
Figure 5.7 Impact of update interval for different algorithms in Torus network.

even when the latter has small update intervals, and gives the best performance overall.

In HMB and HABBH in cases where there are shared links between candidate paths belonging to different source-destination pairs, then any changes in residual bandwidth on these shared links are conveyed back to all source nodes that have candidate paths that included the shared links. In this way any staleness of QoS information is avoided.



(a) Flow blocking probability



(b) Bandwidth rejection probability

Figure 5.8 Flow and bandwidth blocking probability in Rand80.

### Impact of range of load states

Figure 5.8 represents the HMB and HABBH algorithms supplemented with the disjoint and recalculation methods in the Rand80 topology network. The Figure

evaluates the output of the new algorithms against CBR and WSP by calculating the flow blocking probability which is graphed versus a range of loads. Figure 5.8(b) shows the bandwidth rejection probability of the same algorithms versus a range of different loads. Both Figures show that the blocking probability approaches 0 under the load of 0.3. With the load over 0.3 the blocking probability steadily increases for all algorithms, whereas the bandwidth rejection steeply increases because of the eventual residual bandwidth utilization. The Figure 5.8 shows that use of HMB and HABBH notably decreased flow and bandwidth rejection probability in this particular type of the network.

Two factors usually affect any routing mechanism. These factors are (1) the main plan of the routing, which is a global or localised algorithm, and (2) the path selection method. In case of the WSP algorithm, the path is chosen according to the regularly updated global state of information.

The performance of WSP is seriously compromised if the link state updates do not keep up with the traffic fluctuations in the network. Thus, the WSP selection method always opts for the most suitable path based on the information provided by the last update, and this may be very different from the actual (current) link state if the update interval is long. Consequently, WSP performs as the worse scheme among all the algorithms (Figure 5.8) for large update intervals.

Our algorithms and CBR share the approach of localised routing which does not need any network update but they differ in the path selection way. The preferred path for the CBR algorithm is the one with the maximum credits until it rejects the flows. The credits are updated after every flow, based on the status of the chosen path. Once a rejection has occurred, an alternative, higher credited path is chosen. Both HMB and HABBH select the best feasible path based on the source view which gets the clear observation of the network to select the optimum path by taking in consideration the bandwidth as the key QoS metric.

Therefore, the main feature of the algorithms in assessing their performance is the estimation of the flow blocking probabilities as in the Figure 5.8. To match flow blocking probability to an optimum bound we used WSP with update interval 0, so that WSP then has instantaneous updates plus a global view, as shown in the Figure 5.8. Clearly this therefore gives a lower bound on blocking probability but anything approaching it is unattainable in practice. The performance of our proposed algorithms constantly falls between that of WSP (0) and WSP (30), with some algorithms being comparable to the performance of WSP (0). The WSP algorithm with the update interval 30 performs the worst because of its extended update interval of the global state. The CBR technique is already superior to the WSP (30) due to the alternative path selection implemented in its principle of routing. Nonetheless, the flow blocking probability used to credit the paths in

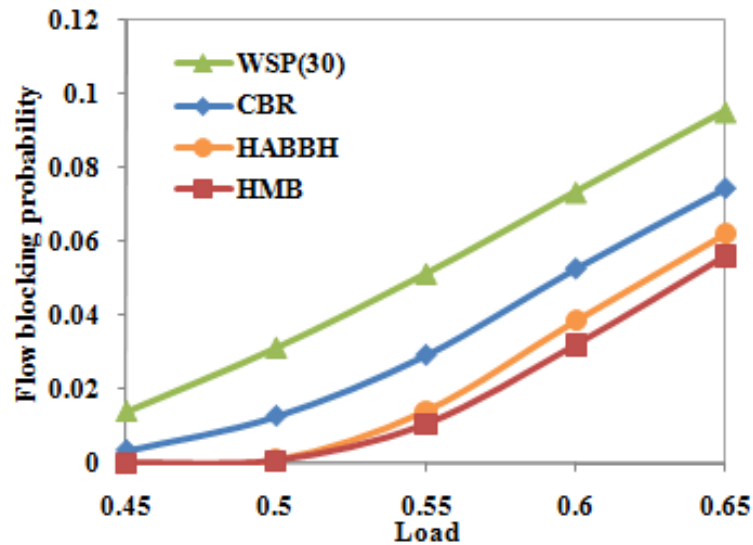
CBR is still not as efficient as the use of the bandwidth metric chosen to quantify the paths in our suggested algorithms.

The experiment demonstrates that both new algorithms perform more successfully for all types of networks with the best performance given by HMB, which is described in the following section.

### **Impact of Network topology**

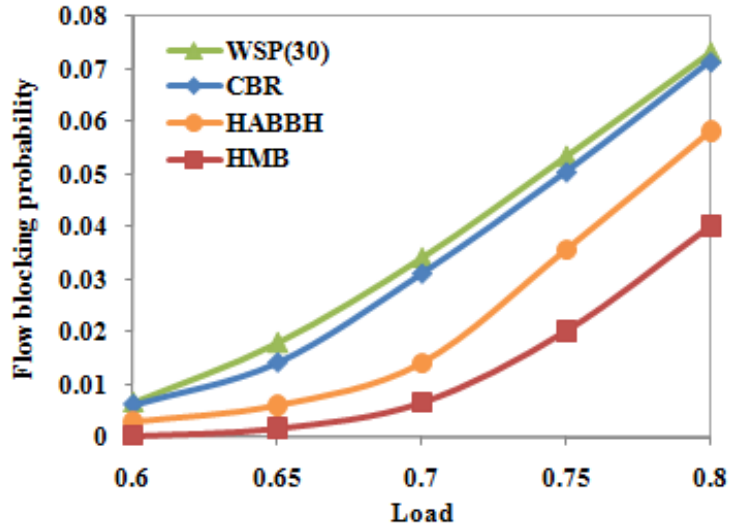
Any routing algorithm performance is radically conditioned by the diverse types of network topologies. Therefore, there is a need to examine the performance of our proposed algorithms with both CBR and WSP under different network topologies. Figure 5.9 shows the effect of the network types, mentioned above in Table 1, on the performance of the algorithms evaluated by the level of flow blocking probability. The Figure shows that the superior performance of the HMB algorithm is not significantly affected by the type of network topology apart from the Torus network case. Figure 5.9(d) represents the HMB and HABBH algorithms and compares the output of the new algorithms to WSP with update intervals of 30 seconds and the CBR by calculating the flow blocking probability graphed versus a range of load states in the Torus network topology. The use of HMB and HABBH using local information notably decreased flow blocking probability compared to WSP, even with small update intervals in both regular and random topology networks. Although WSP performs better than CBR in the

Torus network presented in Figure 5.9(d), this is most likely due to the uniformity of traffic, since the Torus network is a more regular topology which tends to reduce the route flapping; both our algorithms outperform WSP and give superior results. The graphs illustrating the performance of the algorithms also show superior performance of HMB and HABBH in all tested types of networks under the presented range of loads. Thus, these results suggest that both HMB and HABBH perform better than CBR and outperform WSP in all different network topologies examined.

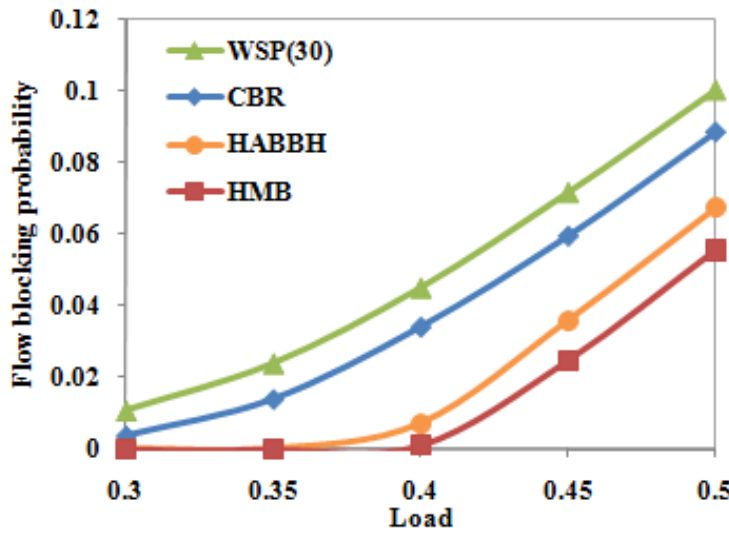


(a) ISP topology

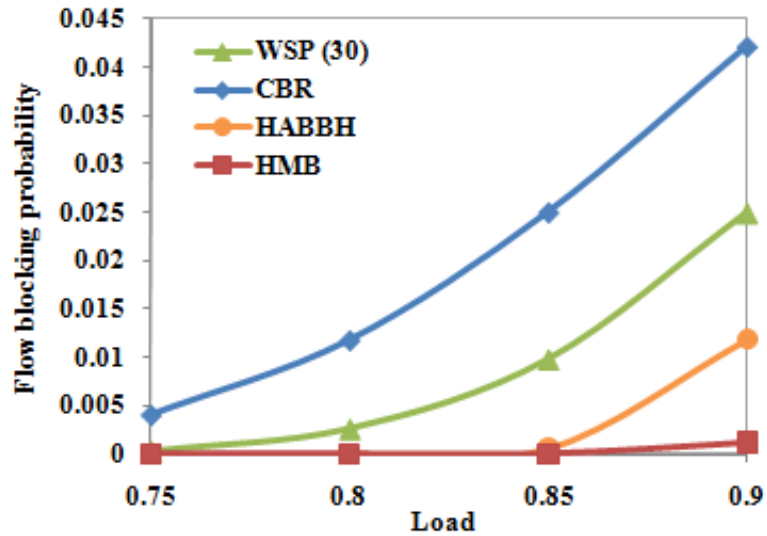
Figure 5.9 Flow Blocking Probabilities in different topologies



(b) Rand45 topology



(c) Rand80 topology



(d) Torus topology

Figure 5.9 Flow Blocking Probabilities in different topologies (Continued)

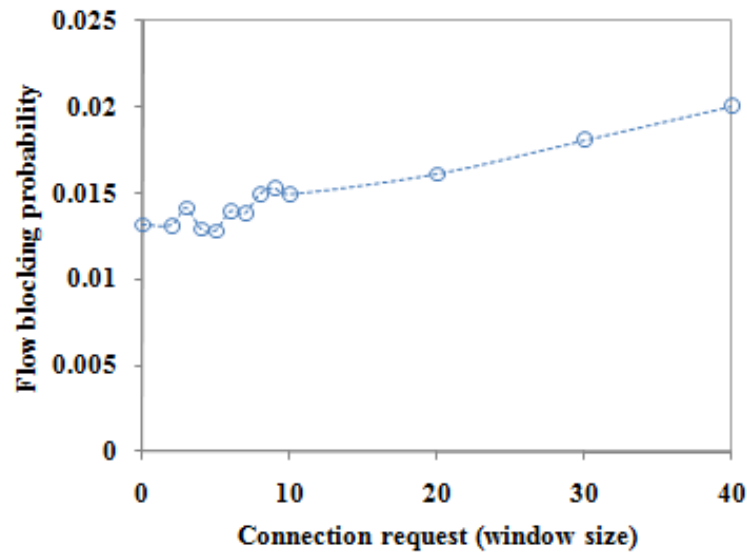


Figure 5.10 Impact of  $w$  parameter in Rand80 topology

### HABBH sensitivity to $w$ parameter

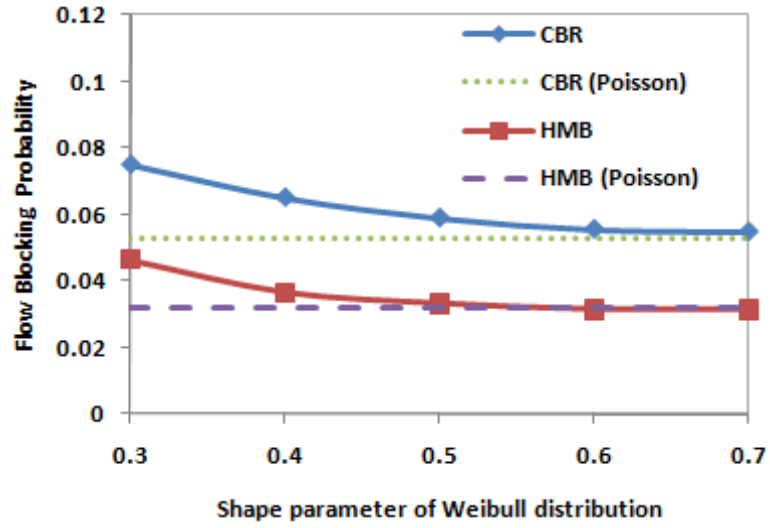
In HABBH a sliding window of size  $w$  observes and records the residual bandwidth in each link of the candidate path sets. This parameter ( $w$ ) is defined as a fixed period of connection requests which is predefined by the network administrator.

Figure 5.10 shows the blocking probability plotted against window size for Rand80, with a fixed load of 0.45. The performance of HABBH is evaluated by using different values of  $w$  connection requests. It is apparent from this Figure that selecting long periods of connection requests leaves a poorer choice for the paths to be selected; therefore the HABBH algorithm performs worse. In other words, the blocking probability increases as the value of  $w$  increases. HABBH gives

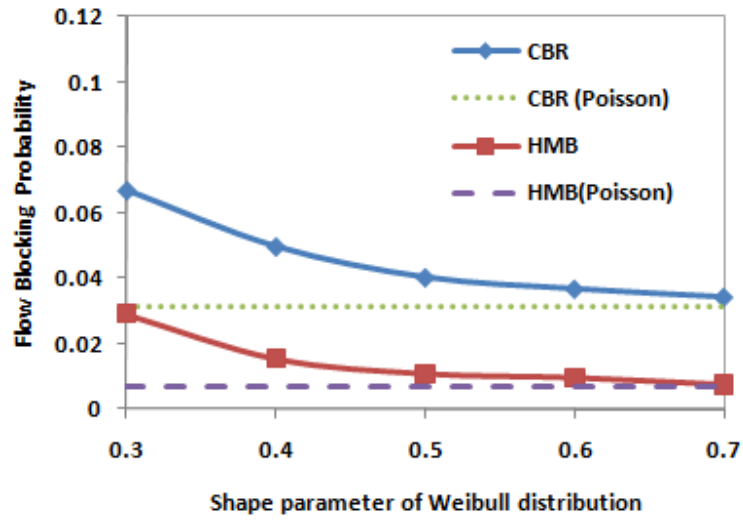
better performance with a value the window size lower than 10. The smaller size of the window, from 0 to 5, does not generally disturb the excellent performance of the HABBH algorithm by increasing the blocking probability and allows selecting the best path among the candidate paths in the set. The Figure 5.10 shows that values up to 5 fluctuate the blocking probability and the values of 6 and upwards gradually increase the blocking probability as the window size increases.

### **Impact of Bursty Traffic**

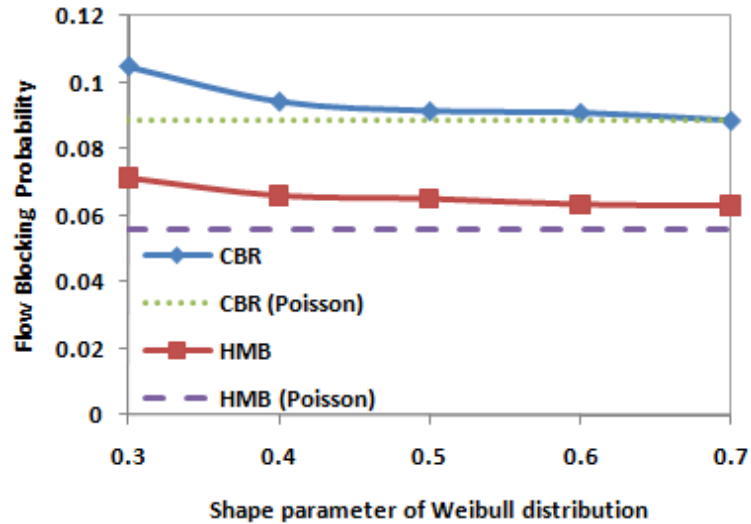
As presented in [67, 74] we now analyse the performance of the proposed algorithms HMB and CBR under bursty settings. The experiment is carried out using various flow lengths according to a Weibull distribution with shape parameters from 0.3 to 0.7. The burstiness is increased with a small shape value. Figure 5.11 shows the flow blocking probability versus different shape values for different algorithms on four network topologies. The amplified burstiness in the arrival process causes an increased blocking probability over the range of shape parameters used. Although the 0.3 shape parameter worsened the performance of both algorithms, HMB performance was the least affected by the burstiness of traffic, since its routing decision is taken based on the bandwidth as the QoS metric unlike CBR which employs blocking probability which is elevated with burstiness.



(a) ISP topology

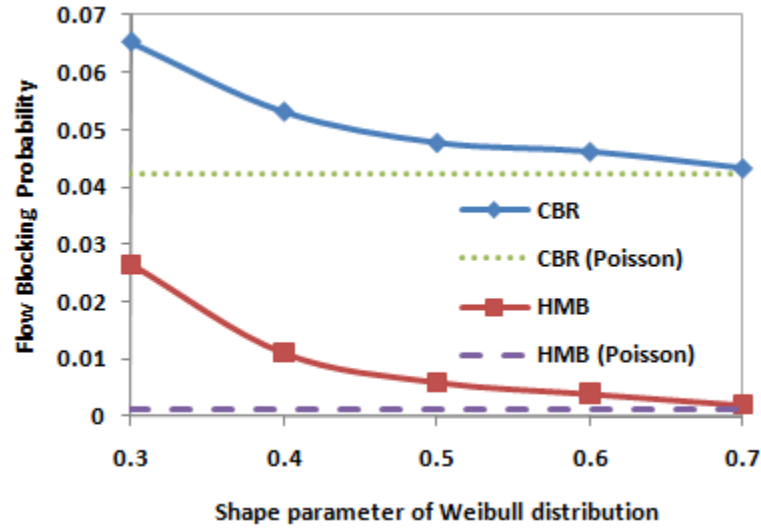


(b) Rand45 topology



(c) Rand80 topology

Figure 5.11. Impact of bursty traffic on different algorithms in different network topologies

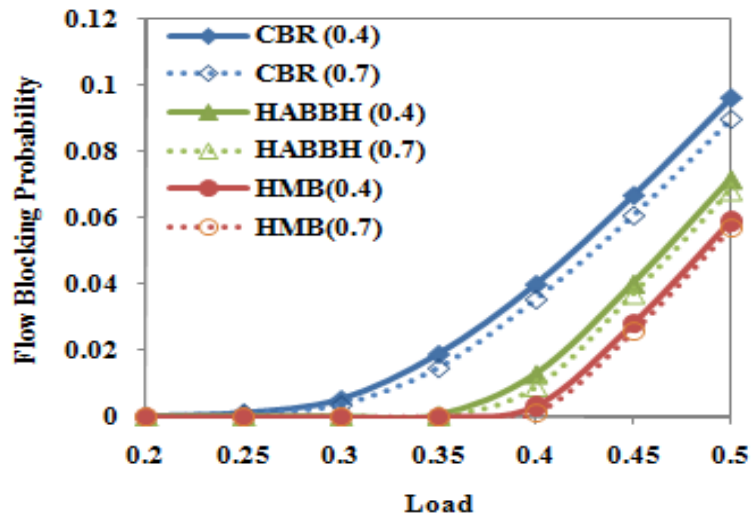


(d) Torus topology

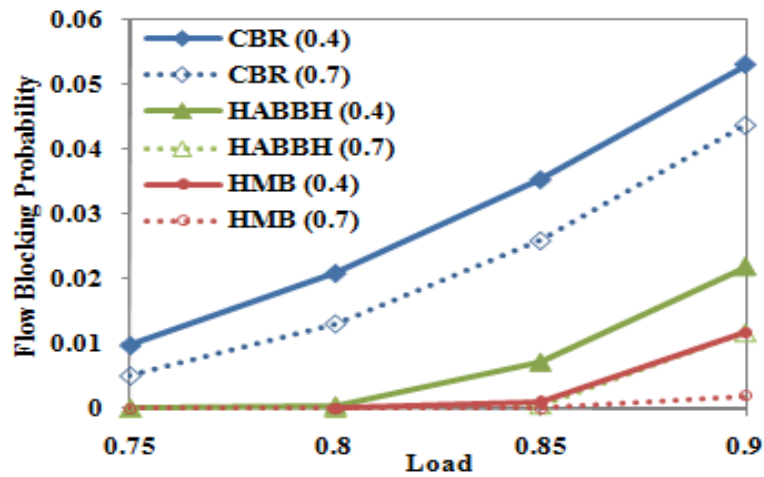
Figure 5.11 Impact of bursty traffic on different algorithms in different network topologies (Continued)

Figure 5.12 represents the effect of bursty traffic on networks of regular (Torus) and irregular (Rand80) topologies. Figure 5.12(a) shows the flow blocking probability versus the offered load with different shape values, where small shape parameter equals 0.4 and large shape parameter equals 0.7, for the random topology network Rand80. The amplified burstiness in the arrival process causes an increased blocking probability over the range of loads used. Although there is no major impact of burstiness in the Rand80 topology, the other set of the experiments in Figure 5.12(b) demonstrate that burstiness has considerable effect on the performance of CBR in the Torus topology, whereas the effect of

burstiness on HMB and HABBH can be seen only at higher loads. This is due to the fact that the new algorithms use bandwidth as a QoS metric directly, unlike CBR which uses the blocking probability in its crediting scheme and which is only updated when a path is used, and this may be infrequently.



(a) Rand80 topology



(b) Torus topology

Figure 5.12 Impact of Bursty traffic for different network topologies

## Routing Scheme Overheads

Both path selection and collection of information are the major reasons leading to overhead in the network. Selecting of a path in global QoS algorithms is based on finding the shortest path by applying some variant of Dijkstra's algorithm or Belman-Ford's algorithm. In localised routing algorithms, the preferred path is selected from a set of the candidate paths  $R$  by a localised mechanism. Therefore, most of the global routing algorithms take no less than  $O(N \log N+L)$  time to select the path, where  $N$  refers to the network size measured by the number of the nodes. The complexity in finding and setting up the path in localised algorithms is  $O(H)$ , where  $H$  is the average length of the candidate paths in hops. This is because in setting up a path each hop of the chosen candidate path needs to be traversed by the setup message.

The overheads accumulated during the application of any QoS routing scheme can be classified into update overhead and path computation overhead. The principle of the global QoS routing uses a variant of Dijkstra's algorithm. It seeks the whole network for the optimum path, which is the shortest path with the maximum bottleneck bandwidth. Contrarily, localised QoS routing selects the candidate path from the candidate paths set( $R$ ) with the time complexity of  $O(H)$ , which is significantly better than the  $O(N \log N+L)$  in global schemes. In global QoS routing, gathering of the information state requires an exchange of the link state

information among the routers in the whole network. This difficulty leads to other ones, such as flapping of routers and high communication overheads. On the other hand, localised QoS routing is considered a viable alternative method to global state routing which can provide better performance. The information collected locally and provided to the network QoS state by the source nodes minimizes the communication overhead and eliminates the need for the routers to update and keep a database of the QoS state.

Therefore the result of the localised QoS routing schemes is the reduction of the communication overhead in the network and simplification of the routing mechanism giving more scalability over the global QoS routing schemes.

## **5.5 Summary**

Although there are similarities in the main ideas of localised QoS routing algorithms shared between CBR and our presented algorithms, there are a number of major differences. CBR routes the flow based on a crediting scheme that rewards a successful path and penalizes the failed ones. Furthermore, CBR credits the whole path as one block, whereas our algorithms choose a suitable path on the basis of the best link from compared ones. We use a totally different technique in both of the algorithms. Residual bandwidth in each link plays a fundamental role in all our algorithms.

In this chapter we offered two methods to improve the performance of the CBR algorithm and introduced new localised routing algorithms HMB and HABBH. We analyzed their performance compared to CBR and WSP in different network topologies. In four types of networks, ISP, Rand45, Rand80 and Torus, our algorithms consistently performed better than both CBR and the global routing algorithm WSP for realistic update intervals.

The methods offered for selecting the candidate paths, which are disjoint paths, recalculation paths and the dynamic selection method, not only improved the function of the CBR algorithm but allowed the proposed algorithms to perform more beneficially. We analyzed the performance of the two proposed algorithms with both the disjoint and recalculation path methods. The HMB algorithm generally gave the best performance and decreased blocking probability the most.

# **Chapter 6**

## **Integrated QoS Routing and Call Admission Control Algorithms Using a Localised Approach**

### **6.1 Introduction**

Communication delay, as well as bandwidth limitation, still stands as one of the key metrics of network performance. The end-to-end delay has become a challenging subject in networking projects and estimation and increasing demand on many QoS susceptible applications, such as stream media, motivates the managing of the end-to-end delay metrics. Contemporary internet applications offering real-time services are only available if the delay constraint is met correctly. Thus, the understanding of the end-to-end delay can lay the foundations of Service Level Agreement (SLA) validation between network service producers and consumers [105].

The significance of the delay guarantee for networks has been outlined by contemporary research, with the particular interest in mean delay. This delay category is characterised as an average of all monitored values within a specific

time period [106, 107]. Although this cannot guarantee instantaneous delay, as might be required by some strict real time applications such as safety critical ones, it provides a useful metric for services in which prompt delivery is important.

In order to overcome known disadvantages of global QoS routing, we have proposed competent localised QoS routing algorithms and presented them in the previous chapter. Even though, the proposed localised algorithms express simplicity in selection of feasible paths, providing a linear time complexity  $O(H)$  with expected small protocol complexity, they still do not cater for real-time applications with specific end-to-end delay constraints.

This chapter we dedicate to the description of four localised QoS routing algorithms taking their routing decisions based on mean end-to-end delay as the QoS metric. The implemented algorithms have been evaluated in terms of flow blocking probability under different network loads and in different network topologies against the global shortest path QoS routing scheme (Dijkstra) [57]. We have also matched them against High Path Credit (HPC), the modified CBR algorithm proposed in [4], which has been modified to use delay instead of bandwidth and performs routing decisions based on flow statistics of path blocking probability.

## 6.2 Admission Control

Integrated Delay Based Routing and Admission Control (IDBRAC) is a mechanism which is presented in this chapter. Our proposed algorithms are all localised QoS routing approaches which are based on the IDBRAC scheme. In this section we will show the advantage of using this new technique and describe its methodology.

We assume the delay metric is mean delay since to use instantaneous delay as a QoS metric would not be meaningful. With bandwidth the admission control is done automatically due to bandwidth being a link based metric. That is, if any link has insufficient residual bandwidth to satisfy a requested connection request then the connection request is rejected. Therefore this does not impact on the existing connections. Delay, as a path based additive metric, needs to ensure that any new connection request does not jeopardise the QoS delay of existing connections, which is a major requirement and can be problematic for localised algorithms. In what is presented, the admission control is integrated with the localised routing such that it is also carried out in a localised way.

A call admission control mechanism in localised routing can operate as follows: As a first step, the accumulated delay in the selected path needs to be equal to or less than the QoS delay constraint. That is,

$$\sum_{l \in L} delay(l) \leq QoS\_Delay\_Constraint, \quad (6.1)$$

Here,  $L$  is set of links in the selected path. The second stage involves applying the admission control, which needs to check that accepting the new flow will not affect and jeopardise any existing flows. This can happen when existing paths have shared links with the new connection request in the selected path. If the new path includes existing flows, the shared link delays are added to their accumulated delay and the new delay value is compared to the QoS constraint for these paths. In case of any of them being jeopardised by exceeding the QoS delay requirement for that path the new connection request will be rejected.

Figure 6.1 shows the flow chart of the call admission control in IDBRAC. For instance, to illustrate this mechanism in Figure 6.2 we will assume that  $(n1)$  is the source and  $(n5)$  is the destination for the new connection request and the selected path for this flow will be  $(n1-n3-n4-n5)$ . An existing connection request will be between source  $(n2)$  to destination  $(n6)$  along the path  $(n2-n3-n4-n6)$ .  $X$  and  $Y$  represent link delays in the network.  $QoS_1$  and  $QoS_2$  are QoS delay constraints of both new connection request and existing connection request respectively. To accept the flow and move to next step in the algorithm, the total link delay,  $(3X + Y)$ , needs to be equal to or lower than  $QoS_1$ . The following step requires the algorithm to check the existing connection request is not jeopardised by ensuring

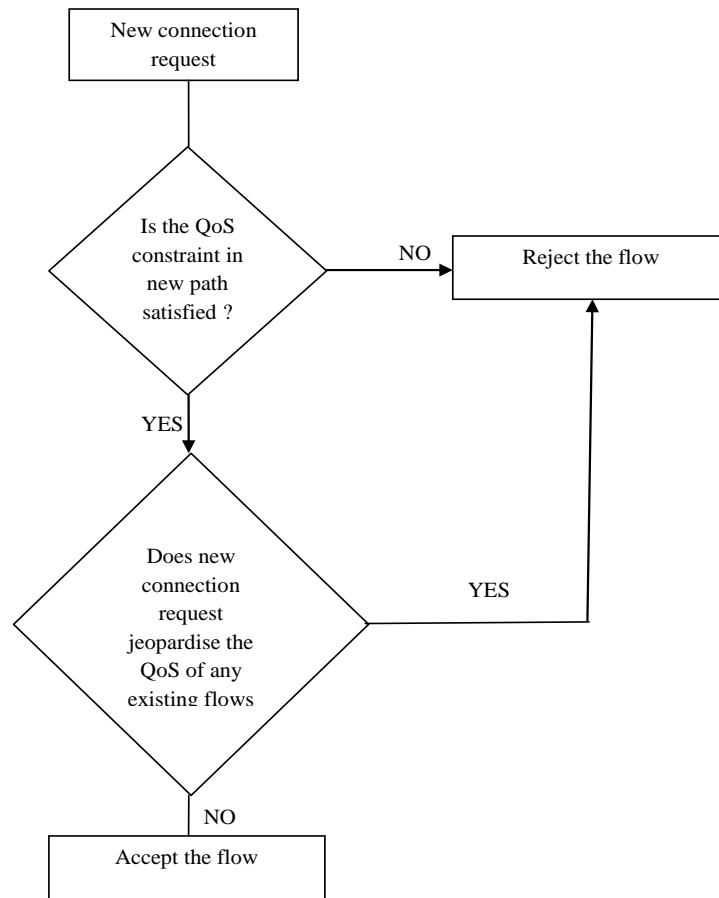


Figure 6.1 Flow chart of Call Admission Control algorithm

that  $(3Y + X)$  is equal or lower than  $QoS_2$ , otherwise the new connection request will be rejected.

It should be made clear, that we are here talking about the mean delay and not the instantaneous delay since the later would need to be represented by a random variable which has a probability distribution and so changes from instant to instant. This would make it meaningless as a QoS metric. By using the IDBRAC

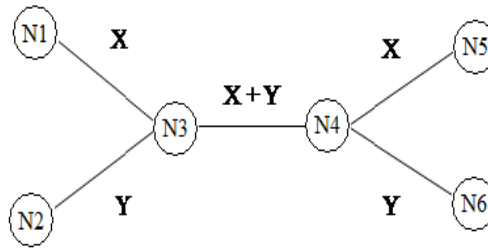


Figure 6.2 Call Admission Control algorithm.

mechanism we present four different localised algorithms. Description of the methods of how the set of candidate paths is chosen is given in the next section.

### 6.3 The Proposed Algorithms

In the present section we illustrate four suggested localised routing algorithms. They are projected as source routing algorithms since the source node's function is to select a path to the destination node that satisfies the QoS. The underlying assumption of the proposed algorithms is the resource reservation and forwarding of signalling messages to establish feasible paths is carried out by the network. The key QoS metric used in our proposed algorithms is mean end-to-end delay. The algorithms provide for traffic flow between source and destination nodes with a mean end-to-end delay which never exceeds a specific assigned value.

### 6.3.1 High Path Credits (HPC)

HPC establishes a setup message which travels from source to destination along the outgoing links in the selected path. The message serves to accumulate the delay over the outgoing link. Each intermediate node runs an admission test for the outgoing link and sums up the outgoing link delay with the preceding delay. Accumulated delay in the selected path needs to be equal or less than the QoS delay constraint. It is the first step before accepting the new flow, see equation (6.1). The second stage is applying the admission control, which needs to check that accepting the new flow will not affect and jeopardise any existing flow. The method investigates the paths which have shared links with the new connection request in the selected path. If these paths already include existing flows, the additional shared link delay is added to their accumulated delay and the new delay value is compared to the QoS constraint for these paths. If any of them are jeopardised by exceeding the QoS delay requirement for that path, the new connection request will be rejected and a failure message will be directed back to the source node.

To process the flow statistics the data about flow acceptance or rejection is collected by the source node. The pseudo code for the HPC algorithm is presented in Figure 6.3.

```

Initialize
  Set P.credits = MAX_CREDITS,  $\forall P \in R$ 
HPC ( )
1.   If P.credits =0  $\forall P \in R$ 
2.   Set P.credits = MAX_CREDITS,  $\forall P \in R$ 
3.    $P_{min} = \max \{P.credits: \forall P \in R_{min} \}$ 
4.    $P_{alt} = \max \{P.credits: \forall P \in R_{alt} \}$ 
5.   If  $P_{min}.credits \geq \Phi \times P_{alt}.credits$ 
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 selected path &
     $\sum \{L'.delay + L.delay : L' \in P' \text{ and } L \in P\} \leq (QoS\_Delay)$  For all
    existing paths  $P' \forall P' \cap P \neq \emptyset$ 
11.  UpdateBlockingProbability (P)
12.  Amount = (1- P.BlockingProbability (P) )
13.  P.credits =  $\min \{P.credits + \text{amount}, \text{MAX\_CREDITS}\}$ 
14.  Else
15.  UpdateBlockingProbability (P)
16.  Amount = (P.BlockingProbability (P) )
17.  P.credits =  $\min \{P.credits - \text{amount}, 0\}$ 

```

Figure 6.3 The pseudo code for the HPC algorithm

Attention should be paid to step 10, where the second term in the AND clause serves to guarantee that QoS of any existing path  $P'$  that shares links with the requested path  $P$  is not jeopardised by the new connection.

A set of candidate paths  $R$  between each source and destination pair is required in the HPC algorithm. Like PSR, HPC predetermines a minhop path set  $R^{\min}$  (line 3) and an alternative path set  $R^{alt}$  (line 4), where  $R = R^{\min} \cup R^{alt}$ .  $P.credits$  is attributed to MAX\_CREDITS. It is a system factor associated with each candidate

path  $P \in R$  to establish and store the maximum credit. HPC selects the largest credit path,  $P.credits$ , in each set (the minhop path set  $R^{min}$  and alternative path set  $R^{alt}$ ) upon flow arrival. The flow is routed along the minhop path with the largest credit  $P^{min}$ , provided  $P^{min}$  is larger than the alternative path that has the largest credit  $P^{alt}$ . Otherwise, the flow is routed along an alternative path, if the following condition is satisfied.

$$P^{min}.credits < \Phi \times P^{alt}.credits, \text{ where } \Phi \leq 1 \quad (6.2)$$

The symbol  $\Phi$  is a system parameter that controls the usage of alternative paths. If the QoS delay constraint value satisfies the end-to-end delay of the selected path and the delay constraint of any existing path is not breached (line 10), the flow is accepted. Path credits ( $P.credits$ ) are increased and decreased upon flow acceptance and rejection respectively using the blocking probability of the path (line 11-17).

The HPC 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.

The main problem with HPC is that a path's credits are only updated as many times as that path is selected. If a path is selected infrequently, then its credit

value becomes stale leading to errors in the selection process. To overcome the mentioned drawbacks we offer some alternative localised algorithms.

### **6.3.2 Minimum Total Delay (MTD)**

The Minimum Total Delay (MTD) method is the second new algorithm offered for localised QoS routing. It uses the total delay in the path to perform its routing decisions. MTD greatly differs from PSR, CBR and HPC by the technique of path selection, which employs the actual delay statistics of each candidate path. Consequently, computation of the total delay for each candidate path is exploited to match the quality of the path and the path with the least total delay is used to direct the arriving flow. The MTD algorithm constantly screens the entire network and updates each path's total delay in the candidate path set.

The process begins by selecting a path to direct a flow with the arrival of a new connection request at a source node. The source node calculates the path that satisfies the specified QoS delay by sending a setup message along the selected path with each connection request. Each intermediate node completes an admission test for the outgoing link and sums up the outgoing link delay with the preceding delay. If the delay is less than or equal to the QoS delay, it is passed to the next node and the delay is reserved for the flow. The flow is accepted if the delay of the selected path is equal to or less than the QoS constraint, as in (line 3).

The flow is rejected by a failure message directed back to the source node if the delay over the selected path is above the QoS delay, or the delay constraint of a present path is exceeded. The latter case indicates that the delay over that path does not match the delay constraint. The source node analyses the flow statistics using data about flow acceptance or rejection.

Remarkably, some authors do not recognise the mean delay as a QoS metric; however, we classify it as such in our work since although we cannot offer any guarantee that the actual delay will not exceed a specific value, it is useful for services that require prompt delivery. The pseudo code for the MTD algorithm is given in Figure 6.4.

```

Initialize
  Set Ptotal = 0,  $\forall P \in R$ 
MTD ( )
1.   Set P = min {Ptotal:  $P \in R$ }
2.   Route flow along path P
3.   If sum {L.delay:  $L \in P$ }  $\leq$  QoS_Delay for selected path &
sum {L'.delay + L.delay :  $L' \in P'$  and  $L \in P'$ }  $\leq$  (QoS_Delay) For all
existing paths  $P' \forall P' \cap P \neq \emptyset$ 
4.   Flow accepted
5.   Calculate Total delay (P)
6.   P.delay = sum {L.delay:  $L \in P$ }
7.   Ptotal = (P.prevDelay + P.delay)
8.   Else
9.   Fow rejected

```

Figure 6.4 The pseudo code for the MTD algorithm.

The MTD localised routing algorithm is programmed so that each source-destination pair predetermines a set of candidate paths  $R$ . In a set of candidate paths every path  $P$  is associated with its mean end-to-end delay as a MTD related key metric.  $P_{total}$  is a cumulative value of the total end-to-end delay and is updated with every connection request. The path with the least total delay (line 1) is chosen and an arriving flow is routed along the selected path. On its way to the destination the setup message adds the delays of the outgoing links of each node in the path and checks that the specified delay constraint is still satisfied after each hop (lines 2-3). Each node along the selected path stores the delay of each flow and uses this data to run an admission control for each new connection request to ensure that any existing flow does not have its QoS endangered by the new connection. In case when the flow is allowed along the selected path, the end-to-end delay is an estimate along that path, and the path total delay is summed up with the preceding (delay) values of the path and stored in the source node (lines 5-7). The probability of the path to be selected for new connection drops with the rise of the total path's delay. Alteration in the total path delay qualifies the real path state, which can be precisely defined. The MTD algorithm relies on the total delay of a path stored in the source node for the acceptance or rejection of a flow on each path.

### 6.3.3 Low Fraction Failure (LFF)

The failure in the path is implemented in the routing decisions of this algorithm. The LFF method involves fraction failure statistics in each candidate path to select the preferred path. The LFF algorithm computes the fraction failure of each candidate path and the preferred path is selected as a path with the least fraction failure to transmit the incoming flow. Similar to The MTD algorithm, LFF screens the delays in the network, but provides constant update of each path's fraction failure in the candidate path set. Thus, fraction failure is used to determine the quality of the path.

The arrival of a new connection at a source node initialises the signalling process in order to forward the flow along the selected path, which satisfies the QoS delay requirements. It is completed by the setup message sent along the selected path with each connection request. The message, started at the source node, accumulates the delay passing through each node and outgoing link and is processed by each intermediate node. Each intermediate node runs an admission test for the outgoing link summing the queuing delay and outgoing link delay with the previous delay. If the delay carried by that message is less than or equal to the QoS delay, the link is tentatively reserved for that flow and the message is passed to the next node. The flow is accepted, if the delay accrued in the complete selected path is less than the QoS delay, as in (line 3). If the flow is rejected a

failure message is returned back to the source node; that is, if the delay over the selected path is above the QoS delay, or the delay constraint of any existing path is exceeded. The statistics concerning flow acceptance or rejection are processed by the source node. The pseudo code for the LFF algorithm is presented in Figure 6.5.

```

Initialize
  Set Pfrac = 0,  $\forall P \in R$ 
LFF ( )
1.   Set  $P = \min \{Pfrac: P \in R\}$ 
2.   Route flow along path  $P$ 
3.   If  $\sum \{L.delay: L \in P\} \leq QoS\_Delay$  for selected path &
 $\sum \{L'.delay + L.delay : L' \in P' \text{ and } L \in P\} \leq (QoS\_Delay)$  For all
existing paths  $P' \forall (P' \cap P) \neq \emptyset$ 
4.   Calculate path fraction failure (Pfrac)
5.   PcurrentFrac = number of failure links in the path  $P$  / number of
total links in the path  $P$  which will be always smallest value (0)
6.   Pfrac = (P.currentFrac + Pfrac)
7.   Accept flow
8.   Else
9.   Calculate path fraction failure (Pfrac)
10.  PcurrentFrac = number of failure links in the path  $P$  / number of
total links in the path  $P$ 
11.  Pfrac = (PcurrentFrac + Pfrac)
12.  Reject flow

```

Figure 6.5 The pseudo code for the LFF algorithm.

The LFF localised routing algorithm sets up a predefined set of candidate paths  $R$  in every source-destination pair. Fraction failure of a path is the key attribute associated with every path  $P$  in the candidate path set  $R$ .  $Pfrac$  in our LFF

algorithm is programmed to store the fraction failure of each path and update its value with every connection request. Thus, LFF opts for the path with the least fraction failure (line 1) and permits the flow along the selected path. The setup message reaches the destination adding the outgoing link delay for each hop on its way. Simultaneously, it matches the links along the path with QoS delay to certify this path against the delay constraint (lines 2-3). Each node along the selected path is sanctioned to store the delay of each flow and execute an admission control test for each new connection request to ensure that the QoS of any present flow is not breached by the new connection. When the flow is accepted along the selected path, the value of the fraction failure is assigned to 0, consequently summed with the previous (fraction failure) values of the path and stored in the source node (lines 4-8). The fraction failure accrued along the selected path produces the actual delay that the path is able to tolerate. In case if the flow is rejected the fraction failure is calculated as the ratio of the number of failed links in the path / total number of links in the path (lines 9-12). Therefore, an increase in the path's fraction failure causes a decreased probability of the path being selected for new connections. Change of the path failure fraction reveals the actual path state and permits an estimate of the quality of the path. Thus, Unlike HPC, which monitors flow blocking probabilities, LFF audits fraction failure of a path which is stored in the source nodes.

### 6.3.4 Low Path Failure (LPF)

This algorithm calculates the actual number of failures in the path in order to route the flows. LPF collects statistics about the actual number of failures in each candidate path. Consequently, the failure number in each candidate path is used to determine the quality of the path. The path with the least path failures is granted to direct the incoming flow.

The selection of the preferred path is initiated by the source node with the arrival of a new connection. The source node calculates the path matching the specified QoS delay constraint. A setup message is sent along the selected path accompanying each connection request. Each intermediate node performs an admission test for the outgoing link and supplements the setup message with a new delay value. The message is allowed to the following intermediate node if the delay value is still less than or equal to the QoS delay and the link is tentatively reserved for that flow. The delay of the selected path being less than the QoS delay, as in (line 3), enables the acceptance of the flow. The flow is rejected and a failure message is transmitted back to the source node, if the delay over the selected path is above the QoS delay, or the delay constraint of any existing path is exceeded. The source node retains the data about flow acceptance or rejection to maintain flow statistics. We present below the pseudo code for the LPF algorithm (Figure 6.6).

```

Initialize
  Set Pfail = 0,  $\forall P \in R$ 
LPF ()
1.   Set P=min {Pfail:  $P \in R$ }
2.   Route flow along path P
3.   If sum {L.delay:  $L \in P$ }  $\leq$  QoS_Delay for selected path &
sum {L'.delay + L.delay :  $L' \in P'$  and  $L \in P'$ }  $\leq$  (QoS_Delay) For all
existing paths  $P' \forall P' \cap P \neq \emptyset$ 
4.   PcurrentFail = 0
5.   Pfail = Pfail + PcurrentFail
6.   Accept flow
7.   Else
8.   Calculate path failure (Pfail)
9.   Pfail ++
10.  Reject flow
11.  For each 200,000 connection request
12.  Set Pfail = 0,  $\forall P \in R$ 

```

Figure 6.6 The pseudo code for the LPF algorithm.

As in previously described localised routing algorithms each source-destination pair in LPF uses a predefined set of candidate paths  $R$ . The functional tool linked with every path  $P$  in the candidate path set is the number of failures.  $Pfail$  is assigned to store the number of failures in the path  $P$  and update its value with every connection request. LPF outlines the path with the least number of failures in the path (line 1) to forward the flow along the selected path. On its way to the destination node a setup message accumulates the delays over the outgoing links

for each hop that the message passes. The message matches the accumulated delays on the links along the path to the QoS delay (lines 2-3). Every node along the selected path stores the delay of each flow to complete an admission control for each new connection request so that the QoS of any existing flow is not jeopardised by the new connection. If the flow is accepted along the selected path, the path failure value is equal to the previous value and is stored in the source node (lines 4-6). The number of failures associated with the selected path is used as the criterion to select the path. The rejection of the flow increases the number of the path failures of the candidate path by 1 (lines 7-9). Thus, alteration in the path failure affecting the quality of the path is precisely estimated. In its turn, the path's failure increase reduces its probability to be selected for new connections. The procedure is restarted after completion of each 200,000 connection requests and the data of the previous set is omitted. Considering only the renewed data set avoids bulking the algorithm procedure and makes it practical. A larger number of connection requests used in this window did not significantly affect the results although a small number tended to introduce instability into the results.

## **6.4 Performance evaluation**

In order to benchmark our proposed algorithms, we chose to use the modified CBR, because its accepted performance based on bandwidth as QoS metric [4, 5,

<b>Topology</b>	<b>Nodes</b>	<b>Links</b>	<b>Node degree</b>	<b>Avg. path length</b>
ISP	32	108	3.375	3.177
Torus	49	196	4	3.5
Rand32	32	126	3.936	2.423
Rand80	80	482	6.025	2.99

Table 6.1 The characteristics of the topologies used.

71]. For that reason, we programmed HPC and used it as a standard localised algorithm relying on the delay metric. We also compared our localised algorithms to the global shortest path QoS routing scheme (Dijkstra) [57].

## 6.5 Methodology

The following section illustrates the simulation characteristics and the network topologies used in our experiments.

### 6.5.1 Simulation environment

The localised QoS routing algorithms were structured to operate at flow level. Resource reservation, admission control, and selecting of the preferred path are based on the proposed algorithms and simulated using the discrete-event simulator OMNeT++ [102].

### 6.5.2 Network topologies

We simulate different topologies of both regular and random networks and run our algorithms over these types of networks, because underlying network topology might affect the functionality and efficiency of the routing algorithms. In the chosen random topologies, the Doar-Leslie Model [88] is used to generate these. The Waxman random graph scheme [89] has been modified by adding a scaling factor to produce the Doar-Leslie Model. As an example of a typical regular network [9] [103], in our experiments we used the ISP and Torus topologies. All networks were implemented in OMNet++ combined with C++. Table 6.1 shows the characteristics of the topologies used.

### 6.5.3 Simulation Characteristics

The assumptions applied on our work to simplify the simulation of delay based algorithms are the same as for the bandwidth sustained ones used previously in our work: in all networks, all simulated links are bidirectional. The arrival rate of flows ( $\lambda$ ), directed to each source node, occurs according to a Poisson process. The flow service time is exponentially distributed with mean  $1/\mu$ . Nodes in the networks are either sources or destinations with the source node assigned arbitrarily, and the destination node selected randomly from the remaining nodes. The QoS delay constraint lies between 1 and 3 time units and the mean delay time

is exponentially distributed for each link. All paths between each source-destination pair having a maximum length at most of one hop more than the minimum number of hops are chosen as the candidate paths [5]. In what follows, it is assumed that such a candidate path set is available to all source- destination pairs.

The limitations utilised in the simulation for HPC are MAX\_CREDITS=5 and  $\Phi=1$ . The estimation of blocking probabilities for HPC were conducted based on the most recent 20 flows. The obtained simulation data is based on at least 2,000,000 connection requests (arrivals) and was compiled after the first 200,000 connections requests.

#### 6.5.4 Performance Metrics

A low blocking probability in the network can be the result of an efficient QoS routing scheme. Therefore, the overall flow blocking probability level serves to estimate a QoS routing algorithm's performance. The flow blocking probability is the ratio of the number of flows blocked  $B$  to the total number of flows that arrived at the network  $T$ , as shown in equation (6.3).

$$\text{Flow blocking probability} = \frac{|B|}{|T|} \quad (6.3)$$

## 6.5.5 Simulation Results

### Varying the delay constraints

An extremely large value for the delay constraint greatly increases the number of the paths in the network available to select a path that satisfies the QoS. On the contrary, tight values of the required delay constraints can make the entire network unavailable with the majority of the connection requests being blocked.

Figure 6.7 illustrates flow blocking probability resulting from manipulation by the range of specified delay constraints in different types of network topology. The main features of the networks were previously given in Table 6.1. As it was expected, all algorithms express the lowest flow blocking probability, i. e. gratify the majority of flows under large chosen QoS delay constraint value of 3 time units, because the probability of finding a path satisfying a slack delay constraint is high and most flows are accepted. Conversely, the functioning of the techniques under some constraints may be dramatically worsened by the average path length in the topology. Thus, Figure 6.7 (c) demonstrates, that simulation in the Torus topology with a delay constraint of 3 results in a poorer performance compared to the other topologies. The fact is that the Torus topology has the largest average path length in number of hops conducted across all source-destination pairs. Remarkably, Figure 6.7 (b) shows, superb performance of all algorithms under the

same constraint of 3 in Rand32, as the topology with the smallest average path length. Generally, Figure 6.7 indicates that the blocking probability increases with the tightening of the delay constraint, as expected, and also confirms the difficulty in routing of the flows with small delay requirements.

To estimate the performance of our proposed algorithms, we use the Dijkstra algorithm as a benchmark. The Dijkstra algorithm with update interval 0 gives a theoretical bound on best performance, but is not attainable in practice. Using the Dijkstra algorithm with more realistic update intervals 2 and 0.3 showed relatively poor performance of the global routing due to the staleness of the link state. We chose to present in Figure 6.7 the graphs resulting from Dijkstra's algorithm for ISP and Rand32, as the examples of regular and irregular topology networks. The graphs plotted on Figures 6.7 (a) and (b) have an appearance of a "sandwich", limited by Dijkstra 0 with the bound on performance and Dijkstra (2) in ISP and (0.3) in Rand32 topologies giving the worst results. The performance of our offered algorithms consistently falls in between the two implementations of the Dijkstra algorithm. The performance of HPC was taken to benchmark the proposed algorithms against an existing localised routing algorithm. Best performance is achieved by MTD, followed by LPF results in Rand80 network. In Rand32 the graphs reflecting the MTD and LPF algorithms show a similar trend with slightly better outcome of LPF under tighter delay constraint values.

Similarly, tight delay constraint values, unlike results with higher values, resulted in a marginally improved performance of LPF algorithm compared with MTD. Switching to the Torus network resulted in a poor performance of the LPF compared to the other localised algorithms due to the longer average path length attributed to the Torus topology. Flow blocking probability of LFF and HPC were both similar in all network topologies under the same load.

The reason the LPF gives poor results for the Torus is that in addition to the Torus having longer paths the path history is used to compute the number of failures. Thus the same paths will tend to be used for a longer period of time, even if the instantaneous situation in the network changes.

The MTD on the other hand always takes advantage of the best available path and can switch paths to suit a given situation. Hence its prevalence over LPF in the Torus network.

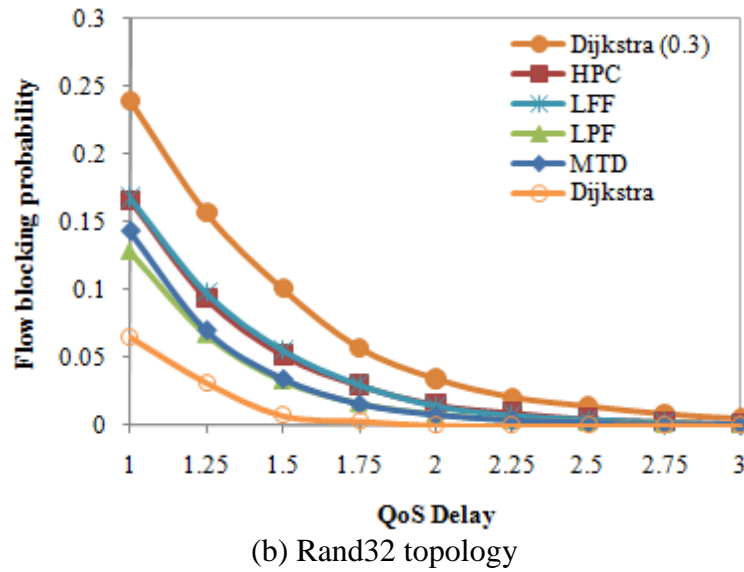
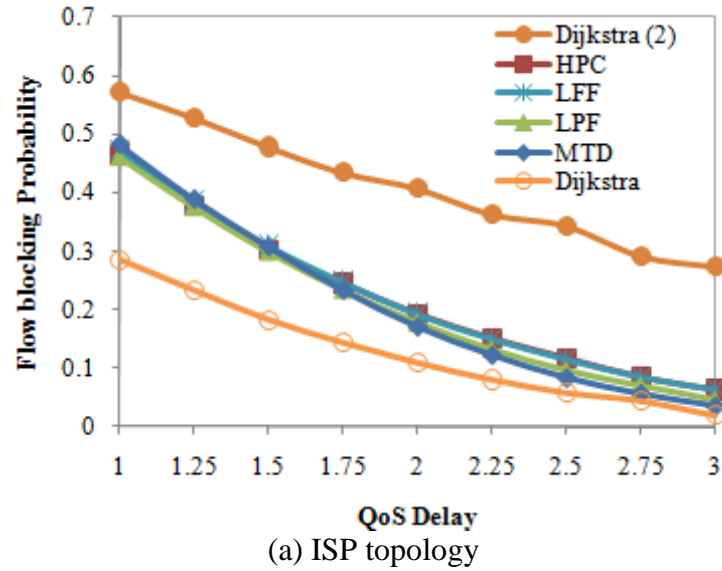
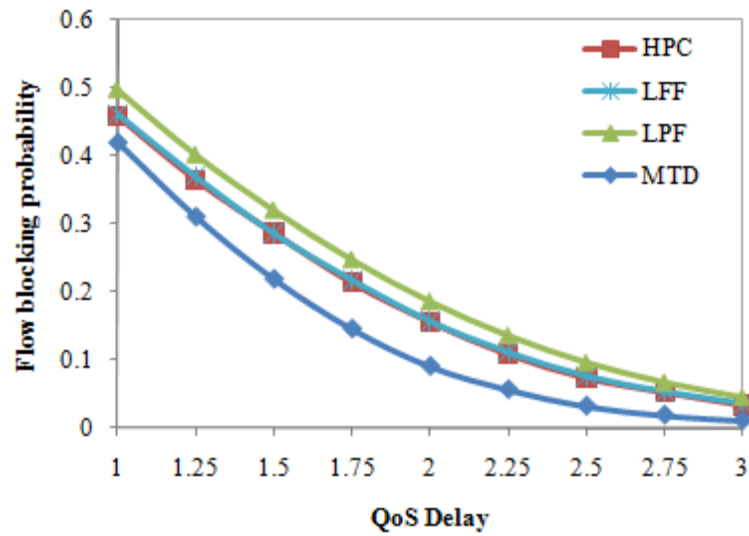
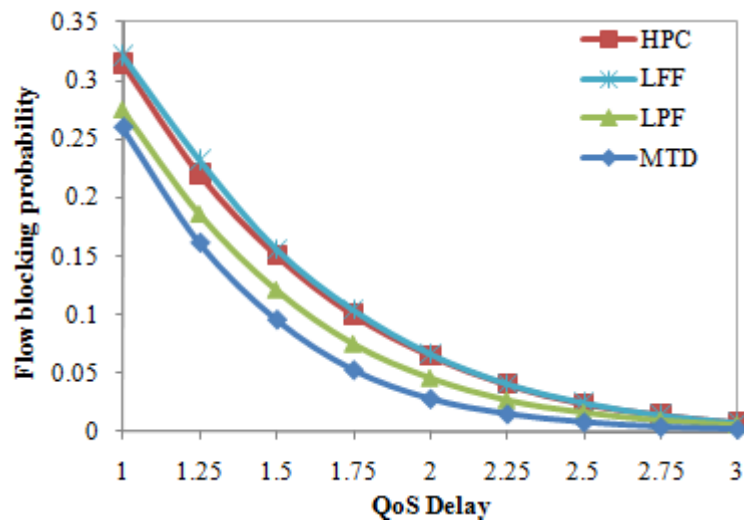


Figure 6.7 Varying delay constraints in different network topologies



(c) Torus topology



(d) Rand80 topology

Figure 6.7 Varying delay constraints in different network topologies

(Continued)

The efficiency of the Dijkstra algorithm is considerably altered by the update interval. It is notable that by increasing the global state information update interval the performance of Dijkstra is significantly worsened manifesting itself in the rapid increase in blocking probability. This is because the longer the interval the staler the link state information becomes. This implies it is also less accurate so Dijkstra with a large update interval is likely operating with inaccurate information. It is also notable that the update interval used is much smaller than the 30 used with bandwidth. This is because an update interval of 30 would mean that Dijkstra (30) would be well off the graphs with the scales used.

As regards the localised routing algorithms HPC, LFF, LPF and MTD, they all act satisfactorily in all network topologies. Their blocking probability values increase steadily with the decrease of the QoS delay constraint. However, the simulation with the same constraints for Dijkstra's algorithm produced a sharp increase of blocking probability. The positive performance of the majority of the localised algorithms occurs due to the effect of alternative routing, which is not based on global state information for path selection. HPC chooses the path with the maximum credits until it rejects the flows, because credits of the selected path are adjustable to blocking probability, and, as a result, selects alternative paths with the updated credit. The moment when a flow is rejected prompts the selection of an alternative more credited path. MTD opts for the paths with the least total end-

to-end delay, which provides better ability to pick paths as long as they satisfy QoS delay. Thus, the MTD, LFF and LPF mechanisms avoid the crediting scheme associated with the HPC scheme by selecting the path based on its quality satisfying QoS delay. Generally, as reflected in Figure 6.7, the path selection method used in MTD provides superior performance of the algorithm, under varying delay constraints simulated in different network topologies opposed to HPC and Dijkstra's algorithms, the latter with small update intervals of 0.3 and 2.

### **Impact of varying arrival rates**

Figure 6.8 reflects the flow blocking probability graphed versus different ranges of arrival rate for different types of network topologies. We begin by using a low arrival rate of 0.5. This allows all algorithms to accept most of the arrivals as long as the average path length is kept small. It is manifested by low blocking probability under 0.3 and 0.1 in case of tight and slack delay constraints respectively. The following increase in rate is used in order to analyse how the local and global algorithms perform. The Figure indicates that in Rand32 and Rand80 topologies, all algorithms for localised routing, such as HPC, LFF, LPC and MTD, express superb performance compared to the Dijkstra algorithm, in spite of a small interval update (0.3) of global state information used. However, it is worth pointing out that the MTD algorithm, which implements path selection

based on the end-to-end delay in a path provides lower blocking probability than HPC in all topologies.

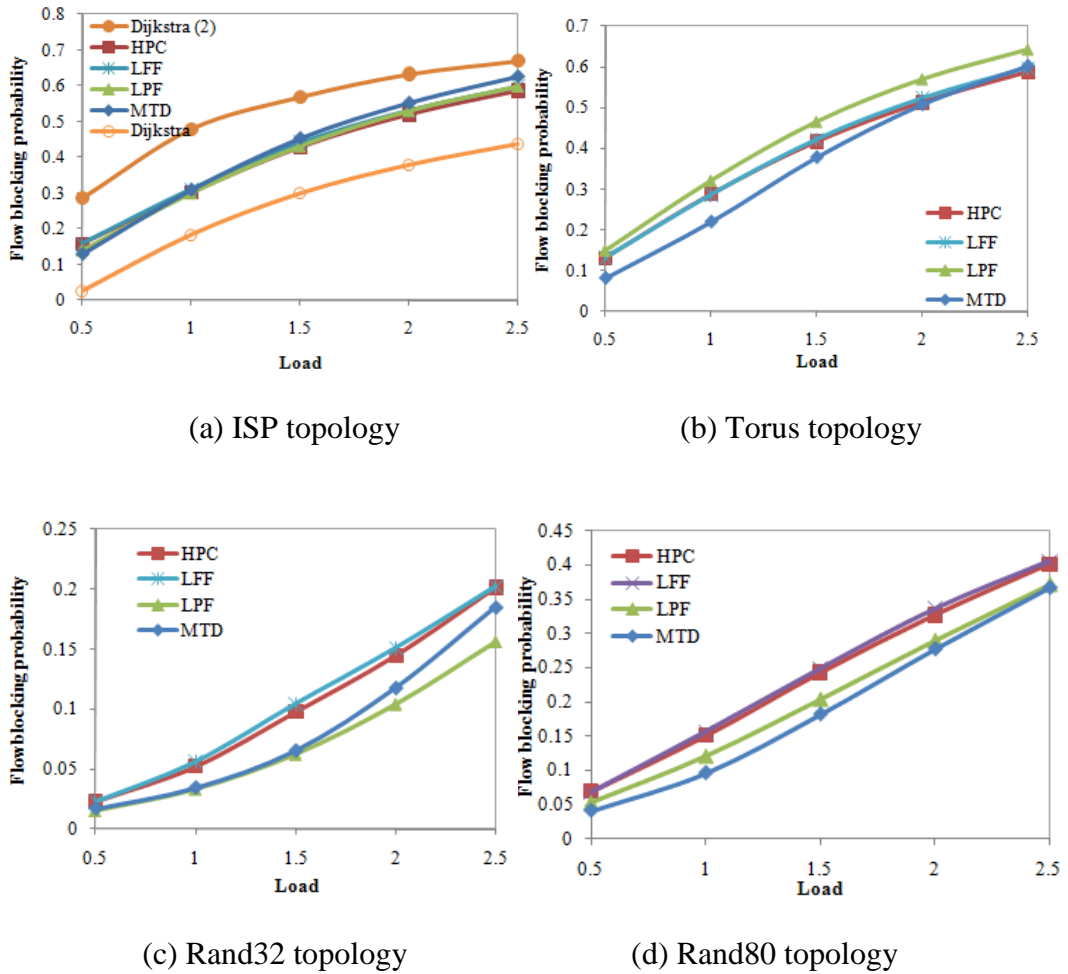


Figure 6.8 varying arrival rates under tight delay constraint (1.5 time units) in different network topologies

Figure 6.8 shows the impact of the changing load between 0.5 and 2.5 with fixed tight QoS delay constraint (1.5) time units. It is worth noting, that the Dijkstra algorithm once again had a similar pattern of a “sandwich” against the rest of algorithms. As the tight values of the required delay constraints make the entire network unavailable with the majority of the connection requests being blocked, all algorithms perform close to each other with no major difference between them. Generally, networks with the irregular topologies, Rand32 and Rand80, produced overall better results, operating with lower blocking probabilities than the ISP and Torus topologies at the same loads. However, performance of offered algorithms had contradicting manner.

Notably, performance of our suggested LFF algorithm consistently corresponded with HPC, being the worst in irregular network types. Thus, in the Rand32 topology, MTD started at the equivalent level with LPF being considerably outperformed by it on the higher loads. The MTD algorithm had a clear prevalence in the Rand80 topology followed by LPF.

In the ISP network, the MTD algorithm gave the worst performance, splitting from the rest of algorithms at a load of 1.5. The same algorithm gave the best results in the Torus network till a load of 2, where its performance corresponded with the HPC and LFF algorithms.

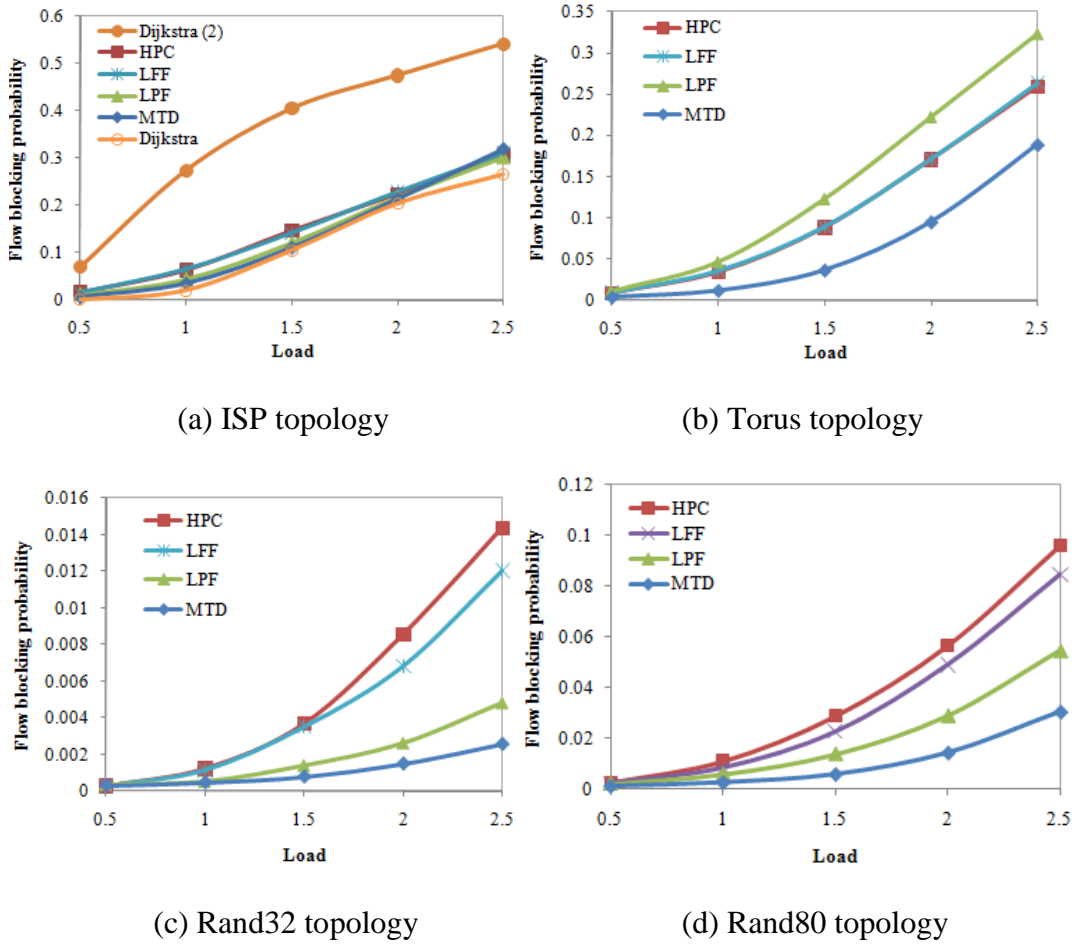


Figure 6.9 varying arrival rates under slack delay constraint (3 time units) in different network topologies

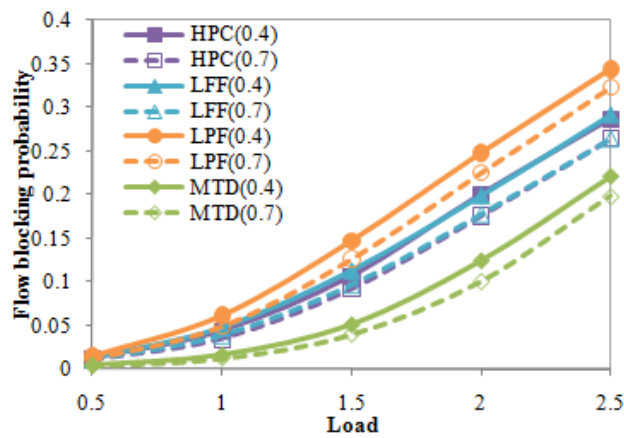
We tested our proposed algorithms with a larger QoS delay constraint equal to 3 under different range of loads and the results are presented in Figure 6.9. Irregular types of networks once again showed an overall lower blocking probability. In the four different network topologies, MTD performed as the best algorithm among the others. LFF was acting the same as HPC in all networks under the same load

(see Figure 6.7). Varying the load gave the LFF algorithm a generally superior performance over HPC in both random topologies, especially Rand80. This occurs due to the fact that random topology gives higher possibilities to select disjoint paths. Apart from the Torus topology, which shows the worst performance for LPF, since the Torus network is a more regular topology which tends to reduce the route flapping, LPF greatly decreased the blocking probability compared to HPC in all other networks and can be considered as the second best algorithm after MTD.

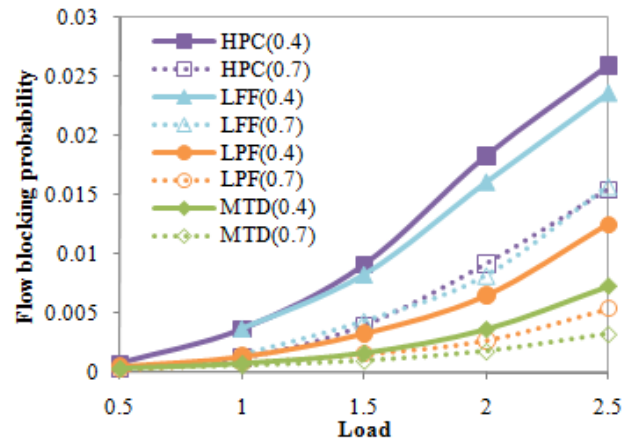
### **Impact of Bursty Traffic**

As presented in [67, 74] we now analyse the performance of the proposed algorithms under bursty settings using a Weibull arrival distribution. Figure 6.10 represents the effect of bursty traffic on network of regular (Torus) and irregular (Rand32) topologies. Figure 6.10(a) shows the flow blocking probability versus the offered load with different shape values, where small shape parameter equals 0.4 and large shape parameter equals 0.7, for the regular topology network (Torus). The amplified burstiness in the arrival process causes an increased blocking probability over the range of loads used. Although there is no major impact of burstiness in the Torus topology, the other set of the experiments in Figure 6.10(b) demonstrate that burstiness has considerable effect on the performance of HPC in Rand32 topology, whereas the effect of burstiness on

MTD and LPF can be seen only at higher loads. This is due to the fact that the new algorithms use mean delay as a QoS metric directly, unlike HPC which uses the blocking probability in its crediting scheme and which is only updated when a path is used, and this may be infrequently.



(a) Torus topology



(b) Rand32 topology

Figure 6.10 Impact of Bursty traffic in irregular and regular network topologies

### **Impact of varying non-uniform traffic**

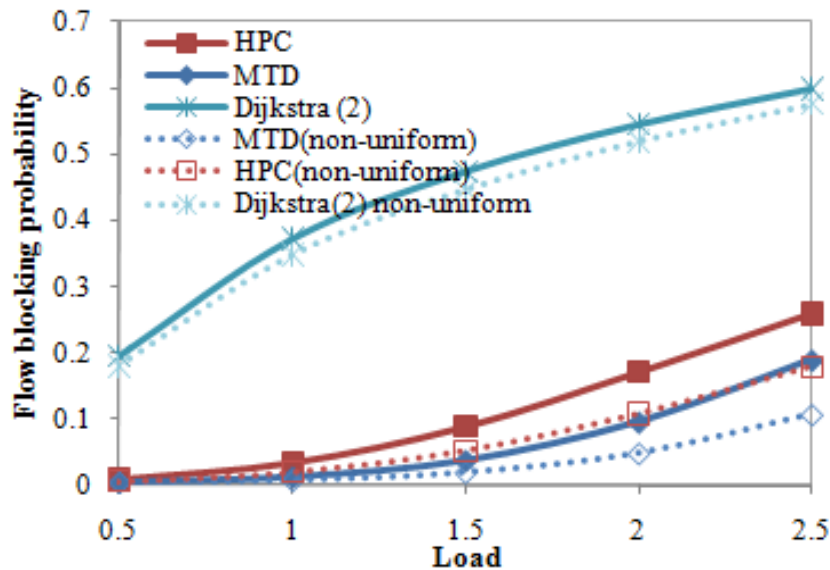
Non-uniform traffic is characterised by the tension on a number of nodes to either receive higher loads of traffic or lower traffic loads than with uniform traffic. The present section describes the effect on our algorithms of non-uniform traffic. We conceptually divided each network, sampled from regular and irregular topologies, into two subnets so that the intensity of the traffic flows routed across the subnets contributed to one third of the overall traffic flows routed within each subnet.

Figure 6.11 represents the flow blocking probability values of selected algorithms<sup>1</sup> plotted versus increasing range of loads under uniform and non-uniform traffic conditions for Torus and Rand32 topologies. Remarkably, the performance of all algorithms under non-uniform traffic was superior compared to uniform traffic in random and regular topologies. This was most likely because the average length of paths become shorter where most traffic was concentrated which gives higher chance of satisfying the QoS metric. In its turn the performance of our suggested algorithms as well as HPC was considerably better than Dijkstra (2). During the application of our algorithms the source nodes have more frequent updates than those provided in the Dijkstra algorithm. Therefore, in

---

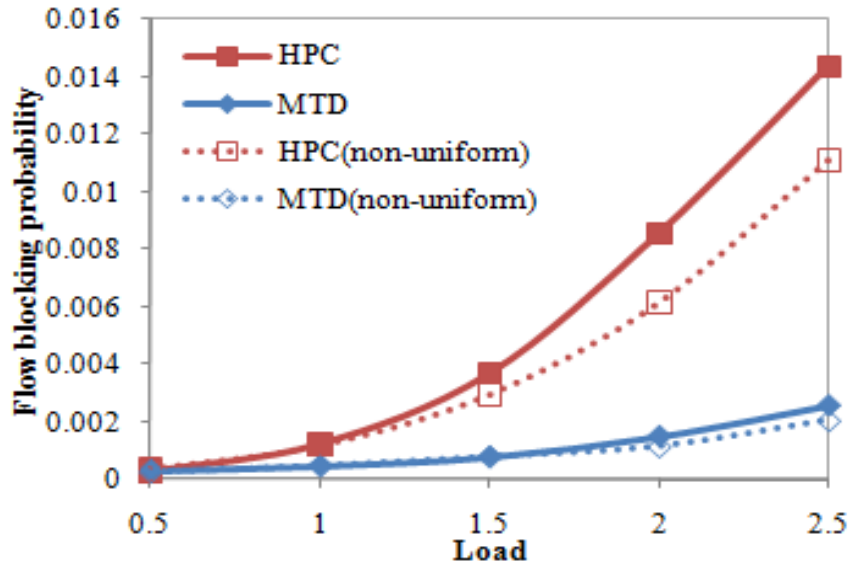
<sup>1</sup> We chose to present in the figure the best algorithm MTD, and compared it to HPC and Dijkstra's algorithm. LFF and LPF performance was similar to MTD and HPC and so is not shown in the graph.

the case of the Dijkstra algorithm, stale QoS updates lead to errors in the route selection. Conversely, since one of the key principles of the localised QoS routing is to collect statistics about network state, implemented localised mechanisms possess more accurate data about frequently used destination nodes with non-uniform distribution. The results in Figure 6.11 demonstrate the very low blocking probabilities that can be achieved by localised algorithms in terms of non-uniform traffic, with no increase in the network overhead. In the Rand32 topology the Dijkstra algorithm result is off the scale.



(a) Torus topology

Figure 6.11 Impact of non-uniform traffic in regular and irregular topologies



(b) Rand32 topology

Figure 6.11 Impact of non-uniform traffic in regular and irregular topologies

(Continued)

## Routing Schemes Overhead

Both path selection and collection of information are the major reasons leading to overhead in the network. Selecting the path in global QoS algorithms is based on finding the shortest path by applying some variant of Dijkstra's algorithm or Belman-Ford's algorithm. In localised routing algorithms, the preferred path is selected from a set of the candidate paths  $R$  by the localised mechanism. Therefore, most of the global routing algorithms take no less than  $O(N \log N + L)$  time to select the path, where  $N$  refers to the network size measured by the

number of the nodes and  $L$  is the number of links in the network. The time complexity in finding the in localised algorithms is  $O (L/H)$ , where  $H$  is the average number of hops in the candidate paths.

## 6.6 Summary

Since the admission control with the delay metric is not automatically done as in bandwidth, we introduced an Integrated Delay Based Routing and Admission Control (IDBRAC) mechanism. Using this technique we offered four localised QoS routing algorithms taking their routing decisions based on mean delay as the QoS metric. The presented algorithms MTD, LFF and LPF were compared with High Path Credit (HPC), the modified CBR algorithm [4], which uses the delay metric instead of bandwidth. The global QoS routing scheme (Dijkstra) [57] was also taken as an evaluative standard. Unlike HPC, which monitors flow blocking probabilities, our localised algorithms monitor the delay of a path directly and the source node stores specific values based on the selected algorithm.

The simulation with the relaxation of the delay constraint resulted in the decrease of blocking probability in all topologies used. Our proposed localised algorithms showed superior performance compared to Dijkstra, even with the latter given small update intervals. Generally, the path selection method in MTD provides

superior performance opposed to HPC being the greatest under varying delay constraints simulated in different network topologies.

Under varying arrival rate, with both fixed tight and slack QoS delay, MTD continued its greater performance among the rest of algorithms in all topologies. The LFF algorithm performed better than HPC in both random topologies, especially Rand80. LPF greatly decreased the blocking probability compared to HPC in all other networks apart from Torus and is considered as the second best algorithm after MTD. The simulation with the non-uniformly distributed traffic shows reduced blocking probability and improves the performance of proposed algorithms. Since high traffic is concentrated in small areas of the network, the candidate paths used for most of the flows would tend to be shorter. Therefore most of the traffic would be directed through these paths which would drop the blocking probability down. It offers great opportunity to apply our algorithms on the internet where traffic flows are very different and thus non-uniform within subnets and in the backbone.

## Chapter 7

# A Distributed Approach to Localised QoS Routing

### 7.1 Proposed algorithm

The algorithm described in this chapter is a distributed routing algorithm, where not only the source node, but also the intermediate nodes take the routing decisions. The suggested distributed algorithm, however, still uses the same strategy of localised routing. The distributed routing algorithm applies residual bandwidth as the direct QoS guideline to select routing paths, unlike CBR which uses the blocking probability as a factor to establish routing paths.

It is an important aspect of our algorithm that by using the metric of interest, such as bandwidth, any changes in this metric due to flows on candidate paths of other nodes can be conveyed back to the source node of all candidate paths involved. In other words, any changes in bandwidth at any node  $x$  which is included in a candidate path  $P$  is relayed to all source nodes which include node  $x$  in one or more of their candidate paths. The fact that all nodes always have up-to-date information concerning their candidate path set makes it a crucial point. Having

up-to-date information would not be possible with CBR [4] or PSR [5] algorithms working with either credit or flow proportions respectively. CBR or PSR would only update the information related to their candidate path sets each time a path from a given set is selected. This means the information could become out-of-date for paths that are only selected infrequently.

## **7.2 Distributed High Bandwidth (DHB) algorithm**

The DHB algorithm differs from the earlier mentioned localised algorithms in several aspects. It selects the highest residual bandwidth among the candidate path set and the mechanism starts when a setup message travels from source to destination along the outgoing link in the selected path. All the outgoing links from the source node in the candidate path set are compared to locate the link with the highest residual bandwidth. In the previous localised algorithms, source routing algorithms, the setup message is directed from source to destination along the outgoing links through one of the candidate paths which have been assigned between the source and destination nodes. In the DHB algorithm, however, after identifying the neighbour node at the end of the link with the highest residual bandwidth among the candidate path set, the algorithm is applied again in this second node to select the link with the highest residual bandwidth in its candidate path set to the required destination. Each intermediate node is thus responsible to

route the setup message through its outgoing link with highest residual bandwidth until the message reaches the destination. In the event of the residual bandwidth being insufficient and so does not satisfy QoS requirements, a failure message is sent back to the source node through the links so far selected informing of the failure of this path. In the successful case, the bandwidth of the message will be reserved from the residual bandwidth and the message will be forwarded until it reaches its destination. The pseudo code for this algorithm is shown in Figure 7.1.

```

Initialize
  Set HighestBW =CAPACITY, P R
DHB ( )
1.   If HighestBW =0 P R
2.   Set HighestBW =CAPACITY,  $\forall P \square R$ 
3.   Set L=max { HighestBW: first L of each P  $\square R$  }
4.   Reserve selected L
5.   While ( next node  $\neq$  destination ) {
6.   Set L=max { HighestBW: first L of each P  $\square R$  } }
7.   Route setup message along L to next node
8.   If flow accepted
9.   Send acknowledgment message back to source through selected links (L)s
10.  Set P= selected links between destination & source
11.  Route flow along P
12.  Else
13.  Free reserved (L)

```

Figure 7.1 The pseudo code for the DHB algorithm

Thus in the DHB algorithm each node in the route selects the best path by choosing the link that has the largest residual bandwidth among the outgoing links of each candidate path from the node to the required destination. In this algorithm a predefined set of candidate paths  $R$  are required by each source and destination pair. However, the source node and each intermediate node need to store the first hop node addresses only to establish the candidate path to destination. In other words, each node acts as a source node by applying DHB algorithm to select the best link, as long as the selected link is a part of the node's candidate path set to the required destination, and so forwards the setup message. A variable, *HighestBW*, stores the highest residual bandwidth link in each node (line 1). The first link of each candidate path in the set  $R$  is compared to each other to obtain the optimum link with highest bandwidth. The selected link is stored in the source node and setup message forwarded to the next node (line 3). The following intermediate node applies the DHP algorithm, acting as a source node, to select the best path of its predefined candidate path set  $R$  by comparing the first link of each path. It is important to mention, that each node needs to store only the first link (first hop) (lines 5-6). As long as the residual bandwidth satisfies the QoS requirements, bandwidth is reserved from the selected link and setup message forwarded to the next node (line 7). When the setup message reaches the destination, notification is sent back to the source node through the selected links

and flow is routed along the selected path P (lines 8-11). If the residual bandwidth is not sufficient and does not satisfy QoS requirements, the flow is rejected and the reserved bandwidth is released (line 13). Importantly, preference has to be given to the link with the highest residual bandwidth among the candidate path set. It thus gives the best chance that the selected link can encompass the flow.

### **7.3 Performance evaluation**

The superior performance of CBR compared to PSR has been shown in [4, 5, 71]. For this reason we again used CBR as a standard localised algorithm to compare our mechanisms.

### **7.4 Methodology**

We used the same simulation environment, OMNeT++ [102], which we used in the previous experiments. Also, the same simulation characteristics that were used in chapter 5 are applied in the DHB algorithm simulation. Network topologies were designed and structured using the same methods as described in chapter 5. Table 7.1 shows the characteristics of the topologies used.

Topology	Nodes	Links	Node degree	Avg. path length
Torus	49	196	4	3.5
Rand45	45	172	3.822	2.692
Rand80	80	482	6.025	2.99

Table 7.1 The characteristics of the topologies used.

### 7.4.1 Performance Metrics

The performance of the distributed QoS routing algorithm (DHB) is estimated by determining the overall flow blocking probability calculated as before and recapped in equation 7.1.

$$\text{Flow blocking probability} = \frac{|B|}{|T|} \quad (7.1)$$

where  $B$  is number of the blocked flows and  $T$  the total number of flows that arrive.

### 7.4.2 Simulation Results

#### Impacts of varying load and network topology

Figure 7.2 evaluates the output of the DHB algorithm against CBR and WSP by presenting the flow blocking probability graphed versus a range of loads in different network topologies.

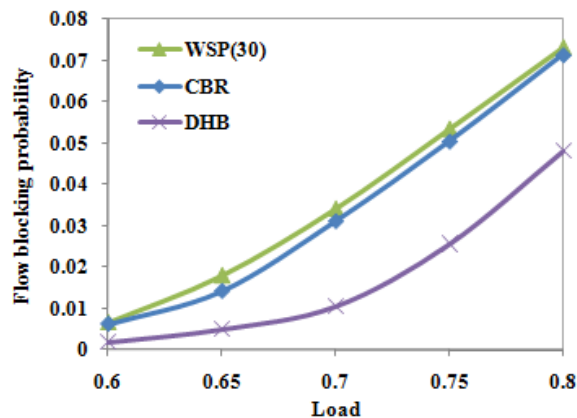
Combining local information and the distributed approach used in the DHB algorithm notably decreased flow blocking probability compared to CBR in both regular and irregular network topologies. This is likely due to the distributed approach, which always takes advantage of selecting best available link and can thus switch paths to suit the present situation when a path is set up.

The DHB algorithm outperforms WSP with the update interval 30, in Rand45 and Rand80, as it is showed in Figure 7.2(a) and 7.2(b) respectively, because of the extended update interval of the global state. The CBR technique is already superior to the WSP (30) due to the alternative path selection implemented in its principle of routing. Nonetheless, the flow blocking probability used to credit the alternative path in the CBR still is not as efficient as the use of the bandwidth metric chosen to directly qualify the path in our suggested algorithm.

The performance of WSP is seriously compromised if the link state updates do not keep up with the traffic fluctuations in the network. Thus, the WSP selection method always opts for the most suitable path based on the information provided by the last update, and this may differ considerably from the actual (current) link state, if the update interval is long. Consequently, WSP performs as the worse scheme among all the algorithms (Figure 7.2) for update intervals (30) apart from the regular Torus network topology.

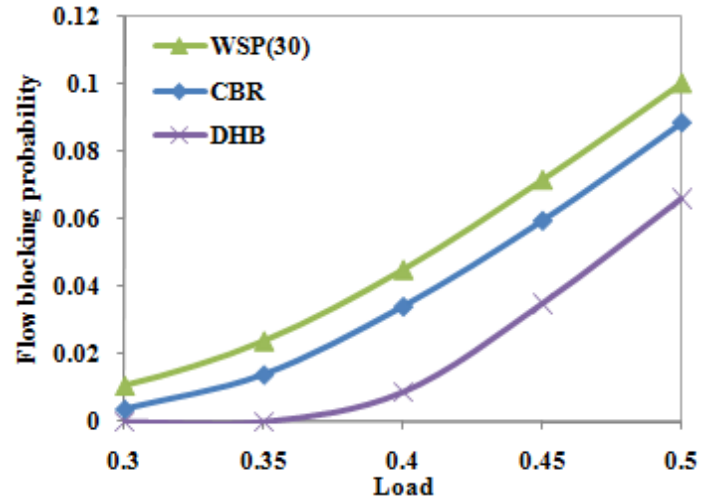
In case of the Torus network Figure 7.2 (c), the simulation showed that the WSP performed better than CBR and DHB. This is most likely due to the uniformity of traffic, since the Torus network is a more regular topology which tends to reduce the route flapping. In addition, the Torus has the longest average path length among the presented topologies. The length of the path could possibly be extended by the distributed approach, which opts for the links with the highest residual bandwidth and thus has the option of switching candidate paths at each hop, which may result in a longer path overall. However, WSP (30) still was the worst in random topologies because of its necessary update interval of the global state.

Thus, these results suggest that DHB performs better than WSP in random topologies and outperforms CBR across all chosen topologies for the simulations conducted.

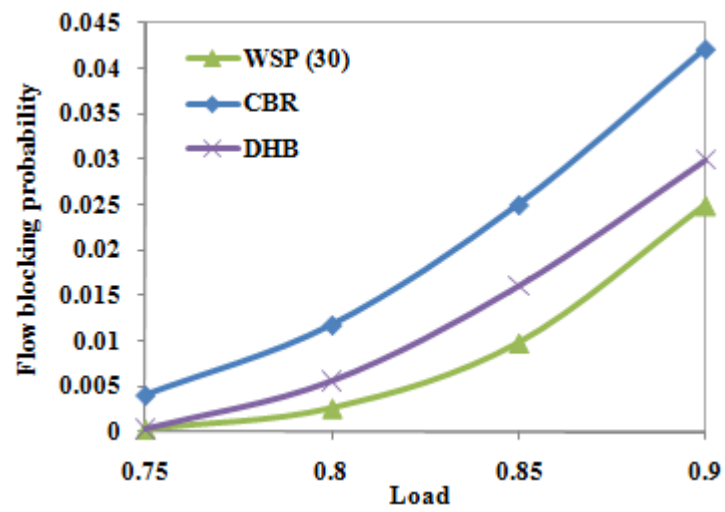


(a) Rand45 topology

Figure 7.2 Flow blocking probability in different network topologies



(b) Rand80 topology



(c) Torus topology

Figure 7.2 Flow blocking probability in different network topologies (Continued)

## 7.5 Summary

The suggested distributed algorithm uses the strategy of localised routing, where the intermediate nodes are able to take routing decisions based on residual bandwidth as the direct QoS guideline to select routing paths. Any changes in the bandwidth metric due to flows on candidate paths of other nodes can be conveyed back to the source node of all candidate paths involved. This avoids the problems of CBR or PSR related to out-of-date information for infrequently selected paths. We used OMNeT++ with the previously used simulation characteristics applied on regular and irregular network topologies and evaluated the DHB algorithm by comparing its performance with CBR and WSP by calculating the flow blocking probability over a range of loads and in different network topologies.

The WSP technique depends on the last update, which may differ from the actual link state if the update interval is long. Consequently, WSP (30) performs the worst in Rand45 and Rand80, because of its extended update interval of the global state. WSP performs better than CBR and DHB in the Torus network due to the uniformity of the topology which reduces the need to switch paths as often as in random topologies. DHB notably decreased flow blocking probability compared to WSP (30) in both Rand45 and Rand80 topology networks.

Thus, the results suggest that DHB performs better than WSP in random topologies and outperforms CBR across all chosen topologies simulated. This

latter occurs due to the distributed approach, which always takes advantage of selecting best available link and can switch paths to suit a given situation.

Both path selection and collection of information are the major reasons leading to overhead in the network. Therefore the localised QoS routing schemes, even with a distributed approach, lead to a reduction of the communication overhead in the network and simplification of the routing mechanism giving more scalability over the global QoS routing schemes.

# Chapter 8

## Conclusions and Future Work

The final chapter is presented as a summary of our research and the conclusions are followed by some suggested future developments.

### 8.1 Summary and Conclusions

- QoS routing strategies have been drawn from the point of view of three different categories, such as number of sources and destinations, state of the information and decision place, contributing to the existing routing algorithms.
- **In its turn** the method of the state information collection can be either global, based on global link state updates among routers in the whole network, or local, based on local path state recordings. An explanation of the global QoS routing and its WSP, SWP, DISP and QOSPF algorithms, as well as the localised QoS routing mechanisms such as DAR, PSR and CBR was given.
- **In order to** discuss and validate the simulator design, we employed several forms of network topologies, parameter conditions and the blocking probability as a measurement metric for performance evaluation.

OMNet++, an open-source communication network simulation environment, has been utilized with the aid of the C++ language to analyse the efficiency of the suggested algorithms.

- Existing localised routing algorithms, CBR and PSR, use the blocking probability as a factor in selecting the routing paths and work with either credit or flow proportion respectively, which makes it almost impossible to have up-to-date information. Therefore our proposed HMB and HABBH algorithms utilise residual bandwidth as the direct QoS criterion to select routing paths. The offered methods for candidate paths selection, i. e. disjoint paths, recalculation paths and dynamic selection method not only improved the function of the CBR algorithm but benefited the proposed algorithms. We analyzed the performance of the two proposed algorithms with both the disjoint and recalculation path methods compared to CBR and WSP in different network topologies. In four types of network topologies our algorithms consistently performed better than both CBR and the global routing algorithm WSP when the latter had a realistic update interval. The HMB algorithm expressed the best performance and decreased blocking probability the most.
- We introduced an Integrated Delay Based Routing and Admission Control mechanism. Using this technique the MTD, LFF and LPF localised routing

algorithms were compared with the global QoS routing scheme, Dijkstra, and modified localised scheme HPC. Our proposed localised algorithms showed superior performance compared to Dijkstra, even when the latter had small update intervals. The simulation with the non-uniformly distributed traffic reduced blocking probability and improved the performance of the proposed algorithms, which offers great opportunity to implement our algorithms in the internet.

- We developed the DHB algorithm applying a distributed approach on localised routing and compared its performance to the localised algorithm, CBR, and the global algorithm, WSP. Simulation results demonstrated a far superior performance for DHB compared to CBR in both regular and irregular network topologies due to adopting a distributed approach, which always takes advantage of selecting the best available link and can switch paths to suit a given situation. In the case of the Torus network, the simulation showed that the WSP performed better than both the CBR and DHB. The Torus network is characterised by the uniformity of topology and longer paths. The length of the path is almost certainly extended in this topology by the distributed approach, which opts for the links with the highest residual bandwidth and in this way thus may create longer paths.

However, WSP (30) still was the worst in random topologies because of its extended update interval of the global state.

Thus, based on our research and existing data, we speculated enough to advocate localised QoS routing as a feasible alternative approach with superior scalability to the global QoS routing algorithms. We motivate our conclusion by the following arguments:

- Overhead resulting from maintaining and distributing the global network QoS state, and the path computation overhead are the source of the scalability challenge in large-scale QoS-enabled networks. Therefore, overhead reduction should become a goal of any developed scaling technique.
- The path selection method in localised routing should rely on such QoS constraints as bandwidth and delay, and mirror the path superiority to be selected or rejected, rather than use indirect metrics, e. g. credit or flow proportion.
- Any changes in bandwidth or delay metrics due to flows on candidate paths of other nodes can be conveyed back to the source node of all candidate paths involved.
- Localised QoS routing maintains a constant set of candidate paths between any source and destination node in the network. However, in a real

network a link failure may occur affecting the network performance. Thus, maintaining a dynamic set of candidate paths has been one of the thesis's concerns.

- The analysis of the multiple simulations illustrated in the thesis demonstrated that network topology, traffic characteristics, load conditions and the strictness of the QoS constraints are all key factors influencing any algorithm efficiency and therefore must be considered for the reasonable appraisal of any QoS routing algorithm.

## 8.2 Future Work

One of the limitations of the localised algorithms is that they cannot be applied directly to hierarchical networks such as the internet. This implies that they would have to be applied separately within the subnetworks and on the backbone. This extension would therefore prove a useful avenue to explore as future work.

Development of the area of study concerned will significantly contribute to further research with the particular interest in extending the localised approach to the case of QoS multicast routing. As far as we know, there is no multicast algorithm based on a localised approach described in available sources. The idea of beneficial performance offered in this thesis relying on bandwidth or delay as the only QoS metric suggests that a delay and bandwidth combination might be

usefully examined in future related work. The method of candidate path selection is another key factor affecting the efficiency of localised routing algorithms and its further development may contribute to improving the performance of the proposed localised algorithms.

Current research suggests that localised QoS routing can be used as a load balancing tool in order to optimize the network resource utilization. Localised QoS routing is characterized by forwarding a QoS connection request along the most feasible candidate path in the candidate path set. The accompanying drawbacks might be overcome by load balancing amongst all the candidate paths of the same candidate path set. The process of load balancing in a candidate path set can involve available bandwidth or end-to-end delay of this path.

The distributed approach suggested in chapter 7 improved the performance of localised routing algorithms and we therefore suggest the development of the distributed approach in localised routing can be a significant avenue for the future work.

## References

- [1] J. Moy, "RFC2328: OSPF Version 2," *RFC Editor United States*, 1998.
- [2] 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.
- [3] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick, "A framework for QoS-based routing in the internet–RFC no. 2386," Internet RFC, August 1998.
- [4] S. Alabbad and M. Woodward, "Localised credit based QoS routing," *IEE Proceedings-Communications*, vol. 153, pp. 787-796, 2006.
- [5] 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, December 2002 2002.
- [6] B. Deng, H. Liu, and M. Zheng, "Overview of QoS Routing," 2008.
- [7] Y. Zheng, W. Dou, J. Tian, and M. Xiao, "An overview of research on QoS routing," *Lecture notes in computer science*, pp. 387-397, 2003.
- [8] Z. Whang and J. Crowcroft, "Quality-of-Service routing for supporting multimedia applications," *IEEE J. Select. Areas Commun*, vol. 14, pp. 1228-1234, 1996.
- [9] Q. Ma and P. Steenkiste, "On path selection for traffic with bandwidth guarantees," in *5th IEEE International Conference on Network Protocols*, Atlanta, GA, 1997, pp. 191--202.
- [10] Z. Wang and J. Crowcroft, "QoS routing for supporting resource reservation," *IEEE Journal on Selected areas in Communications*, vol. 14, pp. 1228-1234, 1996.

- [11] H. Zhu, "Comparison Between Pre-computation and On-Demand Computation QoS Routing with Different Link State Update Algorithms," Citeseer, 2003.
- [12] Lu, s. H. M. K. Costa, S. Fdida, and O. C. M. B. Duarte, "Developing scalable protocols for three-metric QoS routing," *Comput. Netw.*, vol. 39, pp. 713-727, 2002.
- [13] P. V. Mieghem and F. A. Kuipers, "On the complexity of QoS routing," *Computer Communications Journal*, vol. 26, pp. 376-387, Mar 2003.
- [14] S. Siachalou and L. Georgiadis, "Efficient QoS routing," *Comput. Netw.*, vol. 43, pp. 351-367, 2003.
- [15] S. Chakrabarti and A. Mishra, "QoS issues in ad hoc wireless networks," *Communications Magazine, IEEE*, vol. 39, pp. 142-148, 2001.
- [16] W. Pattara-Atikom, P. Krishnamurthy, and S. Banerjee, "Distributed mechanisms for quality of service in wireless LANs," *Wireless Communications, IEEE*, vol. 10, pp. 26-34, 2003.
- [17] C. Toh, "Maximum battery life routing to support ubiquitous mobile computing in wireless ad hoc networks," *IEEE communications Magazine*, vol. 39, pp. 138-147, 2001.
- [18] G. Apostolopoulos, R. Guerin, S. Kamat, and S. Tripathi, "Quality of Service Based Routing: A Performance Perspective, ," *Computer Communication Review*, vol. 28, 1998.
- [19] R. Guerin and A. Orda, "QoS-Based Routing in Networks with Inaccurate Information: Theory and Algorithms," *IEEE/ACM Transactions on Networking*, vol. 7, pp. 605-614, Aug. 1999.
- [20] S. Chen and K. Nahrstedt, "Distributed quality-of-service routing in high-speed networks based on selective probing," 1998, pp. 80-89.
- [21] Y.-C. Hu and D. B. Johnson, "Securing quality-of-service route discovery in on-demand routing for ad hoc networks," in *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks* Washington DC, USA: ACM, 2004.

- [22] 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.
- [23] S. K. Kweon and K. G. Shin, "Distributed QoS routing using bounded flooding," University of Michigan Technical Report CSE-TR388 -99, 1999.
- [24] H. Pung, J. Song, and L. Jacob, "Fast and efficient flooding based QoS routing algorithm," in *IEEE ICCCN99*, 1999, pp. 298--303.
- [25] N. Ouferhat and A. Mellouck, "QoS dynamic routing for wireless sensor networks," in *Proceedings of the 2nd ACM international workshop on Quality of service & security for wireless and mobile networks* Terromolinos, Spain: ACM, 2006.
- [26] F. A. Kuipers and P. F. Van Mieghem, "Conditions that impact the complexity of QoS routing," *IEEE/ACM Transactions on Networking* vol. 13, pp. 717 - 730 August 2005.
- [27] J. SONG, H. PUNG, and L. Jacob, "A Multi-Constrained Distributed QoS Routing Algorithm," in *Proc. ICON 2000*, Singapore, 2000.
- [28] S. Nelakuditi, R. Tsang, and Z. Zhang, "Quality-of-service routing without global information exchange," *Proc. IWQOS 1999*, pp. 129--131, 1999.
- [29] A. S. Tanenbaum, *Computer networks*, 3rd ed. Upper Saddle River, N.J.: Prentice Hall PTR,, 1996.
- [30] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to algorithms*. Cambridge,Mass.; London: MIT Press, 1989.
- [31] C. L. Hedrick, *Routing Information Protocol*: RFC Editor, 1988.
- [32] C. Hedrick, "An introduction to IGRP," *Rutgers-The State University of New Jersey Technical Publication, Laboratory for Computer Science*, 1991.
- [33] B. Fortz and M. Thorup, "Optimizing OSPF/IS-IS weights in a changing world," *Selected Areas in Communications, IEEE Journal on*, vol. 20, pp. 756-767, 2002.

- [34] C. E. Perkins and E. M. Royer, "Ad-hoc On-Demand Distance Vector Routing," in *Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*: IEEE Computer Society, 1999.
- [35] J. Flich, M. P. Malumbres, P. L. pez, and J. Duato, "Performance evaluation of a new routing strategy for irregular networks with source routing," in *Proceedings of the 14th international conference on Supercomputing Santa Fe*, New Mexico, United States: ACM, 2000.
- [36] G. Apostolopoulos and S. K. ripathi, "On reducing the processing cost of on-demand QoS path computation," in *IEEE International Conference on Network Protocols*, Austin, TX, 1998.
- [37] 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.
- [38] P. S. Yu, S. Balsamo, and Y. H. Lee, "Dynamic Transaction Routing in Distributed Database Systems," *IEEE Trans. Softw. Eng.*, vol. 14, pp. 1307-1318, 1988.
- [39] S. Chen and K. Nahrstedt, "Distributed QoS Routing with Imprecise State Information," in *7th IEEE International Conference on Computer, Communications and Networks*, Lafayette, LA, 1998, pp. 614-621.
- [40] W. Lei, S. Yantai, D. Miao, Z. Lianfang, and O. W. W. Yang, "Adaptive multipath source routing in ad hoc networks," in *Communications, 2001. ICC 2001. IEEE International Conference on*, 2001, pp. 867-871 vol.3.
- [41] B. Awerbuch, B. K. Y. Du, and Y. Shavitt, "Routing Through Teranode Networks with Topology Aggregation," in *IEEE ISCC'98*, Athens, Greece, 1998.
- [42] L. Kleinrock and F. Kamoun, "Hierarchical routing for large networks," *Computer Networks*, vol. 1, pp. 155-174, 1977.
- [43] J. C. R. Bennett and H. Zhang, "Hierarchical packet fair queueing algorithms," in *Conference proceedings on Applications, technologies, architectures, and protocols for computer communications* Palo Alto, California, United States: ACM, 1996.

- [44] N. A. Lynch, *Distributed algorithms*. San Francisco, Calif.: Morgan Kaufmann,, 1996.
- [45] P. Lauder, R. Kummerfeld, and A. Fekete, "Hierarchical network routing," *Proceedings of Tri-Comm*, vol. 91, 1991.
- [46] G. Pei, M. Gerla, X. Hong, and C. C. Chiang, "A wireless hierarchical routing protocol with group mobility," in *Wireless Communications and Networking Conference, 1999. WCNC. 1999 IEEE*, 1999, pp. 1538-1542 vol.3.
- [47] C. Jiannong and Z. Fan, "Optimal configuration in hierarchical network routing," in *Electrical and Computer Engineering, 1999 IEEE Canadian Conference on*, 1999, pp. 249-254 vol.1.
- [48] T. Dean, "Hierarchical expectation refinement for learning generative perception models," *Brown University Department of Computer Science Technical Report CS-05-13*, 2005.
- [49] A. Iwata and N. Fujita, "A hierarchical multilayer QoS routing system with dynamic SLA management," *Selected Areas in Communications, IEEE Journal on*, vol. 18, pp. 2603-2616, 2000.
- [50] 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.
- [51] PNNI-Specification-Working-Group, *Private Network-Network Interface Specification Version 1.0*: ATM Forum, 1996.
- [52] Z. Zhang, C. Sanchez, B. Salkewicz, and E. Crawley, "Quality of service extensions to OSPF or quality of service path first routing (QOSPF)," *IETF, Internet Draft*, 1997.
- [53] G. Apostolopoulos, D. Williams, S. Kamat, R. Guerin, A. Orda, and T. Przygienda., "QoS Routing Mechanism and OSPF Extensions," in *RFC 2676*, 1999.
- [54] A. Shaikh, J. Rexford, and K. Shin, "Dynamics of quality-of-service routing with inaccurate link-state information," *University of Michigan Technical Report CSE-TR-350-97*, 1997.

- [55] G. Apostolopoulos, R. Guerin, S. Kamat, and S. Tripathi, "Improving QoS Routing Performance Under Inaccurate Link State Information," in *the 16th International Teletrac Congress (ITC'16)*, Edinburgh, United Kingdom, 1999.
- [56] M. Curado and E. Monteiro, "An Overview of Quality of Service Routing Issues," in *Proceedings of the 5th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2001)*, 2001.
- [57] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, pp. 269-271, 1959.
- [58] D. P. Bertsekas and R. G. Gallager, *Data networks*, 2nd ed. Englewood Cliffs, N.J.: Prentice Hall,, 1992.
- [59] M. R. Garey and D. S. Johnson, *Computers and intractability : a guide to the theory of NP-completeness*. San Francisco: W. H. Freeman,, 1979.
- [60] R. Braden, D. Clark, and S. Shenker, "Integrated services in the internet architecture: an overview," Citeseer, 1994.
- [61] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," Citeseer1998.
- [62] D. Braden, R. Ps, D. Estrin, S. Herzog, and S. Jamin, "Resource reservation protocol (RSVP)--Version 1 functional specification," *Network Working Group, RFC 2209, September*, 1997.
- [63] K. Nichols and B. Carpenter, "Definition of differentiated services per domain behaviors and rules for their specification," RFC 3086, April 2001.
- [64] V. Jacobson, K. Nichols, and K. Poduri, *An Expedited Forwarding PHB*: RFC Editor, 1999.
- [65] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured forwarding PHB group," RFC 2597, June1999.
- [66] X. Yuan, W. Zheng, and S. Ding, "A comparative study of QoS routing schemes that tolerate imprecise state information," *Proceedings of the 11th IEEE International Conference on Computer Communications and Networks*, pp. 230-235, 2002.

- [67] P. Paul and S. V. Raghavan, "Survey of qos routing," in *15th international conference on Computer communication*, Mumbai, India, 2002, pp. 50-75.
- [68] M. Curado and E. Monteiro, "A survey of QoS routing algorithms," in *Proceedings of the International Conference on Information Technology (ICIT 2004)*, Istanbul, Turkey, 2004.
- [69] X. Masip-Bruin, M. Yannuzzi, J. Domingo-Pascual, A. Fonte, M. Curado, E. Monteiro, F. Kuipers, P. Van Mieghem, S. Avallone, and G. Ventre, "Research challenges in QoS routing," *Computer Communications*, vol. 29, pp. 563-581, 2006.
- [70] 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, 1996, pp. 206--217.
- [71] S. Alabbad and M. E. Woodward, "Localized Credit Based QoS Routing: Performance Evaluation Using Simulation," in *the 40th Annual Simulation Symposium*, Huntsville, 2006.
- [72] 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.
- [73] S. Nelakuditi, R. Harinath, E. Kusmierek, and Z. LZhang, "Providing smoother quality layered video stream," in *Proceedings of the 10th International Workshop on Network and Operating System Support for Digital Audio and Video*, 2000.
- [74] 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, 2001, pp. 102-107.
- [75] 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.
- [76] R. J. Gibbens, F. P. Kelly, and P. B. Key, "Dynamic alternative routing - modeling and behavior," in *The 12th International Teletraffic Conference*, Torino, Italy, 1988, pp. 3.4A3.1-3.4A3.7.

- [77] D. Mitra and J. B. Seery, "Comparative Evaluations of Randomized and Dynamic Routing Strategies for Circuit-Switched Networks," *IEEE Trans. on Communications*, vol. 39, pp. 102-116, 1991.
- [78] K. S. Narendra and P. Mars, "The Use of Learning Algorithms in Telephone Traffic Routing - A Methodology," *Automatica*, vol. 19, pp. 495-502, 1983.
- [79] J. T. Moy, *OSPF complete implementation*. Boston: Addison-Wesley, 2001.
- [80] 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, 1997, pp. 115-126.
- [81] R. J. Gibbens, F. P. Kelly, and P. B. Key, "Dynamic Alternative Routing," in *Routing in Communications Networks*, E. M. Steenstrup, Ed. Englewood Cliffs: Prentice Hall, 1995, pp. 13-47.
- [82] *The network simulator ns-2*. <http://www.isi.edu/nsnam/ns/>
- [83] Z. Chengyu, O. W. W. Yang, J. Aweya, M. Ouellette, and D. Y. Montuno, "A comparison of active queue management algorithms using the OPNET Modeler," *Communications Magazine, IEEE*, vol. 40, pp. 158-167, 2002.
- [84] *OPNET Technologies Inc. OPNET modeler website*. [http://www.opnet.com/solutions/network rd/modeler.html](http://www.opnet.com/solutions/network%20rd/modeler.html).
- [85] A. Varga, "OMNeT++ Community Site," in URL:<http://www.omnetpp.org>, 2006.
- [86] A. Varga, "The OMNeT++ Discrete Event Simulation System," in *the European Simulation Multiconference*, Prague, Czech Republic, 2001.
- [87] A. Varga, "OMNeT++ Discrete Event Simulation System Version 3.1 User Manual," in URL:<http://www.omnetpp.org/doc/manual/usman.html>, 2005.
- [88] M. Doar and I. Leslie, "How bad is naive multicast routing?," in *INFOCOM '93. Proceedings. Twelfth Annual Joint Conference of the IEEE Computer and Communications Societies. Networking: Foundation for the Future*. IEEE, 1993, pp. 82-89 vol.1.

- [89] B. M. Waxman, "Routing of multipoint connections," *Selected Areas in Communications, IEEE Journal on*, vol. 6, pp. 1617-1622, 1988.
- [90] L. Wei and D. Estrin, "The trade-offs of multicast trees and algorithms," 1994.
- [91] V. Paxson and S. Floyd, "Why we don't know how to simulate the Internet," in *Proceedings of the 29th conference on Winter simulation* Atlanta, Georgia, United States: IEEE Computer Society, 1997.
- [92] V. Paxson, "Automated packet trace analysis of TCP implementations," in *Proceedings of the ACM SIGCOMM '97 conference on Applications, technologies, architectures, and protocols for computer communication* Cannes, France: ACM, 1997.
- [93] V. Paxson, M. Allman, S. Dawson, W. Fenner, J. Griner, I. Heavens, K. Lahey, J. Semke, and B. Volz, "Known TCP implementation problems," RFC 2525, March 1999.
- [94] R. Pastor-Satorras and A. Vespignani, *Evolution and structure of the Internet: A statistical physics approach*: Cambridge Univ Pr, 2004.
- [95] P. Erdos and A. Rényi, "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci.*, vol. 5, pp. 17-60, 1960.
- [96] M. Newman, S. Strogatz, and D. Watts, "Random graphs with arbitrary degree distributions and their applications," *Physical Review E*, vol. 64, p. 26118, 2001.
- [97] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring ISP topologies with rocketfuel," *IEEE/ACM Trans. Netw.*, vol. 12, pp. 2-16, 2004.
- [98] O. Heckmann, M. Piringer, J. Schmitt, and R. Steinmetz, "On realistic network topologies for simulation," in *Proceedings of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research* Karlsruhe, Germany: ACM, 2003.
- [99] 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.

- [100] S. Chen and K. Nahrstedt, "On finding multi-constrained path," in *IEEE ICC'98*, 1998, pp. 874-899.
- [101] E. Rosen, A. Viswanathan, and R. Callon, "Multi-protocol label switching architecture," *work in progress, Internet Draft draft-ietf-mpls-arch-06.txt*, August 1999.
- [102] A. Varga, "The OMNeT++ discrete event simulation system," 2001, pp. 319-324.
- [103] 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)*, 1998, pp. 80--89.
- [104] 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, pp. 15-27.
- [105] T. Zseby, "Deployment of sampling methods for SLA validation with non-intrusive measurements," in *Proceedings of Passive and Active Measurement Workshop (PAM 2002)*, Fort Collins, CO, USA, 2002, pp. 25-26.
- [106] S. Lee, J. Lui, and D. Yau, "A proportional-delay diffserv-enabled Web server: admission control and dynamic adaptation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, pp. 385-400, 2004.
- [107] M. Crovella, R. Frangioso, and M. Harchol-Balter, "Connection scheduling in web servers," in *Proceedings of the 2nd conference on USENIX Symposium on Internet Technologies and Systems-Volume 2*, 1999, p. 22.