Localised Routing Algorithms with Quality of Service Constraints

Development and performance evaluation by simulation of new localised Quality of Service routing algorithms for communication networks using residual bandwidth and mean end-to-end delay as metrics.

Ding Li

Submitted for the degree of
Doctor of Philosophy

The School of Computing, Informatics and Media
University of Bradford

2010

Abstract

Ding Li

Localised Routing Algorithms with Quality of Service Constraints

QoS routing, localised routing, algorithms, bandwidth, mean delay, simulation

Localised QoS routing is a relatively new, alternative and viable approach to solve the problems of traditional QoS routing algorithms which use global state information resulting in the imposition of a large communication overhead and route flapping. They make use of a localised view of the network QoS state in source nodes to select paths and route flows to destination nodes. Proportional Sticky Routing (PSR) and Credit Based Routing (CBR) have been proposed as localised QoS routing schemes and these can offer comparable performances. However, since network state information for a specific path is only updated when the path is used, PSR and CBR operate with decision criteria that are often stale for paths that are used infrequently.

The aim of this thesis is to focus on localised QoS routing and contribute to enhancing the scalability of QoS routing algorithms. In this thesis we have developed three new localised QoS routing schemes which are called Score Based QoS Routing (SBR), Bandwidth Based QoS Routing (BBR) and Delay Based Routing (DBR). In some of these schemes, the path setup procedure is distributed and uses the current network state to make decisions thus avoiding problems of staleness. The methods also avoid any complicated calculations. Both SBR and BBR use bandwidth as the QoS metric and mean delay is used as the QoS metric in DBR. Extensive simulations are applied to compare the performance of our proposed algorithms with CBR and the global

Dijkstra's algorithm for different update intervals of link state, different network topologies and using different flow arrival distributions under a wide range of traffic loads. It is demonstrated by simulation that the three proposed algorithms offer a superior performance under comparable conditions to the other localised and global algorithms.

Acknowledgements

I am forever grateful to my supervisor, Professor Michael E. Woodward, for his valuable support and guidance throughout my research. I am very honoured to be one of his students, to get his fatherly help, and to learn from him. He leads me to enter the research field of QoS routing. His directions always enlighten me when I meet difficulties in my study.

I thank my parents who encouraged me in every endeavour and taught me to value education. They have been a great support throughout my life.

I would like to thank Miss Rona Wilson, the research administrator, and staff in the student support centre, who provided me with a lot of help throughout my study at the University of Bradford.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

As networks modernize and expand with the increasing deployment of high-speed technology, the Internet has become an important part of people's daily activity. As a packet switched network, the Internet uses gateway routers to interconnect each other. Routing is defined as the process to deliver packets from their sources to the ultimate destinations through intermediary routers, via the most appropriate path. The current Internet protocol offers a single level of service and cannot satisfy real time and multimedia applications, since packets are treated equally which is the so-called the “best-effort” of the Internet. The best-effort networks do not maintain information as each connection arrives at routers, so there is no resource reservation or admission control for each connection. Quality of Service was proposed to re-develop the Internet to satisfy the need of real-time applications. QoS routing is used to find a feasible path that has adequate resources to satisfy a set of QoS requirements of a connection. QoS routing algorithms collect the state of link state information and based on this find a feasible path that satisfies the QoS requirements.

1.1. Motivation

In order to select a path for a flow to meet the flow's QoS requirements such as bandwidth or delay, Quality of Service (QoS) routing requires source nodes to manage some information of the global network QoS state, e.g., the traffic load distribution in the network, and based on this information to make judicious choices in path selection. With expanded networks and the extremely dynamic nature of the Internet traffic, several problems are raised:

(1) For getting accurate link state information of all links in the network at all times, each network node has to keep absolutely up to date information and requires a prohibitively extensive exchange of link state update information between network nodes for all links. The collection of the global network state information consumes an unacceptable amount of network resources and imposes a large communication overhead [1].

(2) If large update intervals are used to reduce routing overhead, stale/outdated information will occur, which may lead to route flapping [1]. When the utilization on a link is low, the out-of-date information causes all nodes to route traffic along this link and results in rapid utilization of this link; otherwise, all source nodes avoid use of the link which has high utilization, and its utilization decreases. This oscillatory behaviour brings on poor route selection, instability and an overall degradation of network performance.

All these problems with existing global QoS routing algorithms become serious issues. Localised QoS routing [2] has been proposed to solve some of these problems, which infers the network QoS state in source nodes based on flow blocking statistics collected locally and perform flow routing using this localised view of the network QoS state instead of using global QoS state information. Localised QoS routing has advantages

over other approaches since the communication overhead and the processing and memory at core routers is greatly reduced. Various localised QoS routing algorithms have been put forward, such as proportional sticky routing (PSR) [2] and the credit based routing algorithm (CBR) [3], but the decision criteria for the paths in these algorithms is only updated when the paths are used and so can also become stale for paths that are only used infrequently. This motivates us to study and develop new localised QoS routing schemes to enhance performance and scalability of QoS routing algorithms.

1.2. Aims and Objectives

The major aim of this thesis is to focus on localised QoS routing algorithms, and to enhance the inherent scalability of QoS routing algorithms. The research sub aims are:

(1) To develop an efficient and scalable localised QoS routing algorithm which selects a path based on residual bandwidth.

(2) To develop an efficient and scalable localised QoS routing algorithm which selects a path using a Random Number Generator with the metric of bandwidth.

(3) To develop a novel localised QoS routing algorithm which selects a path using mean delay as a single metric.

(4) To re-develop the CBR using mean delay as the QoS metric instead of bandwidth.

These aims are to be achieved through the following objectives:

(1) To study, through various literatures, global and local QoS routing and related scalability problems.

(2) To research and understand problems of QoS routing using delay as a single metric.

(3) To develop a simulation platform that can be used to assess the performance of the proposed localised algorithms against other localised and global QoS algorithms.

(4) To develop localised QoS routing algorithms which provide a connection with guaranteed QoS requirement, low message overhead and efficient resource utilization.

(5) To develop efficient and scalable localised QoS routing algorithms using bandwidth as the single metric.

(6) To develop a novel localised QoS routing algorithm that uses mean delay as the QoS metric.

1.3. Original Contributions

The contributions of this thesis can be listed as follows:

(1) We have developed two algorithms which all make routing decisions using residual bandwidth statistics collected locally. One selects a path by a Random Number Generator based on the view that a path with most residual bandwidth has the highest probability to be chosen. Another one selects a path whose residual bandwidth is the maximum.

(2) As a QoS metric, mean delay has some different features. We researched these and noted that the use of admission control is essential. We have modified CBR to meet the requirements for mean delay and have also developed a novel localised QoS routing algorithm which selects a path using mean delay as a single metric and has a distributed path selection process.

(3) We have developed a simulation platform using OMNeT++ and C++ to evaluate performance of our proposed localised QoS routing algorithms. After comparing

them with other localised QoS routing algorithms and a representative global QoS routing algorithm, we demonstrated through simulations that performance of our proposed localised QoS routing algorithms is superior to that of other QoS routing algorithms.

1.4. Outline of Rest of Thesis

The rest of the thesis is organised as follows:

Chapter 2 describes concept of QoS and QoS routing. QoS routing metrics, state information, transmission and decision making are also introduced. We give an explanation about the QoS routing problem and discuss the advantages and limitations of different routing strategies.

Chapter 3 introduces related works of QoS routing algorithms. We explain classes of QoS routing algorithms. We describe global routing algorithms and their shortcoming, and localised routing algorithms and their research evolution.

Chapter 4 expounds the simulation platform we have developed to assess the performance of our proposed QoS routing algorithms. We specify types of network topologies, parameter settings and the performance metrics used in the performance evaluation. The developed simulator and its validation are also described.

Chapter 5 proposes two localised QoS routing algorithms which all make routing decisions using residual bandwidth statistics collected locally. An extensive simulation evaluation of the proposed algorithms and comparison of them with other localised routing methods (CBR) and the global QoS routing (Dijkstra's algorithm) are described.

Chapter 6 describes a novel delay based QoS routing algorithm which is developed using delay as a single metric. We discuss in detail the use of mean delay for QoS

routing and re-develop localised Credit Based Routing (CBR) to meet the environment. They are both evaluated using simulations against the global shortest path algorithm (Dijkstra).

Chapter 7 concludes the thesis and points out possible future directions for the work.

Chapter 2

QoS Routing

2.1. Introduction

The modernisation of the Internet has become an important part of people's daily activity. Business application of the Internet has brought certain challenges in terms of performance and the services offered. Various real-time applications require different levels of service which are not provided by the current Internet which is a so-called “best-effort” network. Quality of Service (QoS) routing is used to solve the problem of how to select a path for a flow such that the flow’s QoS requirements such as bandwidth or delay are likely to be met. In this chapter, we describe QoS, QoS routing and its correlative details.

2.2. Literature Review of Routing

As a packet switched network, the Internet uses gateway routers to interconnect each other. Routing is defined as the process to deliver packets from their sources to the ultimate destinations through intermediary routers, via the most appropriate path. The routing process has two main functions, which are (1) choosing the appropriate paths for various source-destination pairs using routing algorithms such as Dijkstra and

Bellman-Ford's shortest path algorithms [4][5] and (2) delivering messages to their correct destination once the paths are selected, by making use of routing protocols such as Open Shortest Path First (OSPF)[6] and Routing Internet Protocol (RIP)[7].

The packet's header is used to store the information for routers to make a routing decision. The current Internet Protocol (IP) offers a single level of service and cannot satisfy real time and multimedia applications, since packets are treated equally which is the so-called "best-effort" service of the Internet. The routing protocol focuses on the connectivity of network nodes and their links in best-effort networks. The best-effort networks do not maintain information as each connection arrives at routers, so there is no resource reservation or admission control for each connection. The end host needs to provide reliable packet delivery as there is no guarantee of packet delivery. The Transmission Control Protocol (TCP)[8] is used by the end host to retransmit packets when there is no acknowledgment of proper delivery, and the User Datagram Protocol (UDP)[9] is used for applications that rely solely on best-effort.

Although they are simple and scalable, the routing protocols used in best-effort networks have several shortcomings as follows: (1) they primarily use a shortest path routing technique that uses only a single minimum hop path between each source and destination, which can cause uneven distribution of traffic; (2) they do not have admission control and do not differentiate between traffic that uses the network; all connection requests are accepted to the network which can lead to network congestion; (3) packets are sent to the network without delivery guarantee which can lead to packet loss or drops along the shortest path.

2.3. Quality of Service (QoS)

According to the EG 202 009-1[10] “Quality of Service” (QoS) is defined as the collective effect of service performances which determine the degree of satisfaction of a user of a service. It is characterized by the combined aspects of performance factors applicable to all services, such as:

- service operability performance;
- service accessibility performance;
- service retainability performance;
- service integrity performance; and
- other factors specific to each service. [10].

2.3.1. Proposal of Quality of Service

Quality of Service (QoS) was proposed because the modernization of the Internet that has brought it into commercial use has brought about certain challenges in terms of performance and the services offered. Various real-time applications require different levels of service which are not provided by the current Internet which is a datagram model and connectionless network that has imperfect resource management. The datagram model makes use of different routes to send packets of a session to a destination, and packets may be received out of their original order. Some quality of real-time applications such as video on demand and video conferencing is influenced. In addition, alternative paths are not used to route traffic when the main path is overloaded. Now the Internet has become an important part of people’s daily activity, Quality of Service was brought forward to re-develop the Internet to satisfy the need of real-time applications.

Quality of Service (QoS) is also defined as “the collective effect of service performances which determine the degree of satisfaction of a user of the service” [11]. In the Internet, a lot of architectures have been proposed to ensure provision of a QoS protocol, such as the Integrated Services and Resource Reservation Signalling Protocol architecture (IntServ/RVSP)[12][13], the Differentiated Services (DiffServ) architecture[14], the Constraint Based Routing (CBR) [20], the Traffic Engineering (TE)[16], and the Multi-Protocol Label switching (MPLS)[15].

2.3.2. Integrated Service (IntServ)

The Integrated Service (IntServ) is proposed for a per-flow service, which reserves resources such as bandwidths and buffers to ensure the requirements of the given flow getting its requested service. In the network, each device reserve resources for each flow and isolate each flow from the other. With specific QoS for an application flow, Resource Reservation Signalling Protocol (RVSP) is used to reserve network resources and set up the flow. RVSP also performs delivering the QoS requirements to all intermediate routers along the selected path, and recording and maintaining the state of each flow to provide the QoS requested[17]. When a new connection arrives, if the availability of network resources can meet its requirement, the connection is accepted to the network; if not, the connection is rejected. The sender or the receiver of the admitted flow is needed to establish a soft state to manage resources within routers. The soft state is established and periodically refreshed to avoid termination of the flow using RSVP messages. Although it attempts to satisfy the requirement of QoS, IntServ has some drawbacks. When incessant growth of the Internet brings the huge growth of state information, core routers are required to manage very large numbers of flows and this

causes signalling overhead of applying the RVSP protocol.[12]

2.3.3. Differentiated Services (DiffServ)

As the description in Section 2.3.2, the problem in IntServ is that routers have to differentiate between large numbers of connection requests, which result in large overhead to maintain a connection request state table. Differentiated Services (DiffServ) is proposed for solving the scalability problem raised by IntServ. In DiffServ, traffic is divided into aggregated numbers of forwarding classes in the edge router that is responsible for classifying packets into their proper class, so that the complexity in core routers is reduced. The Service Level Agreement (SLA) is used for classification of the packets between service providers. Instead of advance resource reservation setup, Differentiated Services operate the classification of packets on the edge of the network, so that DiffServ is more scalable and flexible. Differentiated Services mark the Internet Protocol (IP) header with the type of SLA applicable, routers then decide how to process the packet when they receive the marked packets. DiffServ has Per Hop Behaviour (PHB), because its networks do not provide the required quality of service, the best-effort treatment is called DEfault per Hop Behaviour (DE PHB). Expedited Forwarding per Hop Behaviour (EF PHB)[19] provides applications with low loss, low jitter, and low latency. Low priority traffic is provided by Assured Forwarding per Hop Behaviour (AF PHB)[18] and delivered packets may be dropped and given low priority in case of congestion. The drawbacks of DiffServ is that it does not provide full guarantees during congestion and does not provide end to end guarantees for real time applications, although it is scalable and offers better performance and less signalling overhead than IntServ/RVSP. [14]

2.3.4. Constraint Based Routing

As a set of protocols and algorithms, Constraint Based Routing facilitates a source node to compute a feasible path to a destination node and consider multiple constraints to increase the utilization of the network[20]. Constraint Based Routing constraints can be treated as administrative costs or application QoS requirements for applications such as bandwidth, delay, jitter and packet loss [21]. Constraint Based Routing takes into account network topology, network traffic requirements and resource availability on links to increase network traffic utilization. The network traffic requirements or constraints can be applied by administrative policies or QoS requirements. Policy constraints are the network traffic constraints applied by administrative policies. Routing based policies are the routing protocols which use this approach. QoS routing is named as the requirement for applications to be satisfied, which is the most important QoS mechanism in the Internet to make the traffic engineering process automatic [22]. Policy routing protocols use paths which match to administrative rules and service level agreements (SLAs). In policy routing, administrators can make routing decisions not only on the destination node location, but also on parameters such as applications used, packet size, or identity of both source and destination end systems.

2.3.5. Traffic Engineering (TE)

To avoid network congestion, the process of managing traffic flows on an IP network is named as Traffic Engineering (TE) [25]. Uneven traffic distribution resulting from using shortest path protocols causes such congestion and overload in some network

links, while others stay underutilized. Traffic engineering aims are to develop and improve network performance through optimization of resource utilization in the network. Traffic Engineering offers an advanced mechanism to allocate traffic in the Internet. The source nodes in the network are required to consider available bandwidth in the network before computing paths. A traffic engineering system should monitor the network topology and network state information and collect any changes that happen to the network due to network dynamics. Network traffic estimation is calculated based on accurate information related to traffic demands of users or calculated based on traffic measurements. The former calculation gets traffic demands from the service agreement between user and service provider. The second calculation gets information from network statistics such as traffic loads on a link and breakdown of traffic types. Route computation calculates routes based on traffic demands, network state information and number of constraints. These constraints can be imposed by routing policies or QoS requirements [24].

2.3.6. Multiprotocol Label Switching (MPLS)

Multiprotocol Label Switching (MPLS)[23] is a new forwarding scheme to solve the problem in routing decisions in IP networks. In MPLS, packets are forwarded based on an attached label. There is a header between the network layer and data link layer in the IP header [24]. A signalling Label Switching Protocol (LSP) is used to set up a path in an MPLS domain, and if the flow is admitted then packets are assigned an MPLS label which determines the path that will be used in that domain. In MPLS headers, Label switch routers (LSRs) are responsible for swapping labels to forward packets on the determined path to their destinations. Since LSR routers efficiently switch the added

labels, MPLS provides fast packet forwarding in large networks. A virtual private network (VPN) can be deployed by MPLS label switched paths [25]. Since the path used by the packet is specified by a single LSR router and should not be carried in the packet header, MPLS can set up an explicit path using a label switched in the MPLS domain. MPLS improves the performance of IP networks by reducing the amount of per packet processing required at each node in IP networks.

2.4. Conception of QoS Routing

QoS routing is the most important QoS mechanism to solve the problem of how to select a path for a flow such that the flow's QoS requirements such as bandwidth or delay are likely to be met. The main objectives of QoS routing are (1) dynamic determination of feasible paths that satisfy the requirements of a flow; (2) network resource optimization and improvement of overall performance by efficient distribution of the traffic in the network and maximization of its resource utilization; and (3) avoidance of congestion hotspots in the network and provision of good performance with heavy loads. For the sake of making judicious choices in path selection, we must have information of the network QoS state such as the traffic load distribution in the network. To develop any QoS routing scheme, two key questions must be addressed: (1) how to get information of the network state and (2) given this information, how to select a path for a flow. In QoS routing, Solutions of these questions influence the performance and cost tradeoffs. [21]

2.5. QoS Routing State Information

The process of QoS routing requires knowledge of the network state information to find a feasible path that satisfies flow requirements which can be measured using certain metrics. With confirmation of metrics, QoS routing state information is used to find a feasible path for a new connection. The methods of collecting the network state information and keeping it up-to-date affect the efficiency of routing. QoS routing state information can be collected based on a global, aggregated or localised approach.

2.5.1. QoS Routing Metrics

QoS routing schemes are required to find a feasible path that satisfies the QoS requirements of a flow. The QoS routing metrics are used to measure the QoS requirements. The QoS requirements can be presented as a single metric or a combination of them. The three most common categories have composition rules which are as follows:

(1) Additive metric: the value of the metric over a path is the sum of the values over each link. The total metric value $w(P) = w(u_1, u_2) + w(u_3, u_4) + ... + w(u_{i-1}, u_i)$, where $u_1, u_2, ..., u_i$ are nodes, path $P = ((u_1, u_2), (u_3, u_4), ...(u_{i-1}, u_i))$. Additive metrics include delay, delay jitter, cost and hop count.

(2) Multiplicative metric: the value of the metric over a path is the product of the values over each link. The total metric value $w(P) = w(u_1, u_2) \cdot w(u_3, u_4) \cdot .... \cdot w(u_{i-1}, u_i)$, where $u_1, u_2, ..., u_i$ are nodes, path $P = ((u_1, u_2), (u_3, u_4), ...(u_{i-1}, u_i))$. Loss probability is a common example of a multiplicative metric.

(3) Concave metric: the value of the metric over a path corresponds to the minimum

value observed in all links of the path. The total metric value $w(P) = \min(w(u_1, u_2), w(u_3, u_4), ..., w(u_{i-1}, u_i))$, where $u_1, u_2, ..., u_i$ are nodes, path $P = ((u_1, u_2), (u_3, u_4), ...(u_{i-1}, u_i)$. Bandwidth is the most widely used concave metric and is also called a link metric. [26]

2.5.2. Global State Information

Global state information is the commonest state information in current networks based on a global view of the information and this is maintained in each node by periodic exchange of link QoS state information among network nodes. Two different protocols are used to maintain and collect the global state in each node: the Distance Vector Protocol and the Link State Protocol. The Distance Vector Protocol uses distance or hop count as its primary metric to select the best path, and periodically exchanges routing tables between neighbouring nodes and assumes that each router knows the distance to its neighbours. Based on a specific QoS metric, the Link State Protocol updates others' information to determine the best path. In the Link State Protocol, link state routers are used to update neighbouring networks with current information, rather than continually providing routing tables to detect change in the state of the routing path [4].

2.5.3. Aggregated State Information

The global network state information becomes difficult to maintain when the network size expands. For solving the problem, a hierarchical topology has been proposed, which clusters the nodes into groups to form logical nodes and clusters logical nodes into groups to form higher level logical nodes, and so on. In each cluster, nodes store

more information about nodes in the same cluster and less information about nodes in other clusters.

2.5.4. Local State Information

Local state information is a key topic in this thesis to research localised QoS routing algorithms. The local state information is collected locally and kept in each node. This local state information can be bandwidth, delay, or any QoS metric. The collected statistics are used as state information to find a feasible path for a new connection [28].

2.5.5. QoS Routing State Information Disposal

While it collects and gets the QoS routing state information, each node in the network uses this information to compute feasible paths. There are three ways for a source node to compute paths: pre-computation [29], on-demand[30] or path-caching [31].

(1) In a pre-computation scheme, each node in the network is required to compute paths for each destination periodically. Each node receives a lot of updates regardless of whether the path is required or not. When a new connection arrives, the updated state information is used to select a feasible path. The pre-computation scheme is better for large networks, and performs well when connection requests are more frequent and can reduce the path setup time [32].

(2) Unlike pre-computation, the on-demand scheme only asks a source node to compute paths to the destination when a new connection arrives. The scheme's strong point is that only the most recent state information is used to compute a feasible path for the new connection arrival. The on-demand scheme is suitable for small networks, and

preferable when arrival of connections are infrequent and less complex as it only calculates a single feasible path.

(3) The path-caching is a hybrid approach of pre-computation and on-demand to reduce the computational cost.

2.5.6. QoS Routing State Information Update Frequency

Since the maintenance of global state information affects the performance of QoS routing, the accuracy and density of QoS routing state information must be considered carefully [1]. An expanding network means that an increasing amount of network resources is consumed by rapid global state information updates for any change in each link state. The rate of updates needs to be reduced so as to decrease the overhead. The OSPF[6], the current widely used protocol, recommends that the link state is updated only once every 30 minutes. This means that it is likely that all link state changes are advertised [33]. Although the large time interval can reduce the overhead, it also leads to stale link state information which can affect QoS routing as follows [34]:

(1) A QoS routing algorithm may not find a feasible path.

(2) A path may be rejected in the setup process because of false information about available link resources.

(3) A QoS routing algorithm may select a non-optimal path.

There are three main link state update policies: periodic, threshold-based and class-based update policies [35][36].

(1) In the periodic policy, the link state information is advertised throughout the network periodically. Since it is not tied to traffic changes, the periodic policy is easy to implement.

(2) In the threshold-based policy, the update is triggered by the relative difference between the available value known by the network and the actual current value of a specific parameter if this exceeds a threshold.

(3) Two classes are used by the class-based policy to divide the QoS parameter. An update of link state is advertised when the actual current value of the QoS parameter changes from lower class to upper class, or vice versa.

2.6. Routing Transmission

QoS routing transmission is generally categorised into three main classes: Unicast, Broadcast and Multicast.

2.6.1. Unicast Routing

Unicast is the term that is used to describe communication where a piece of information is sent from one point to another point. In this case there is just one sender, and one receiver. Unicast transmission, in which a packet is sent from a single source to a specified destination, is still the predominant form of transmission on LANs and within the Internet. All LANs and IP networks support the unicast transfer mode, and most users are familiar with the standard unicast applications which employ the TCP transport protocol.

2.6.2. Broadcast Routing

Broadcast is used to describe communication where a piece of information is sent from one point to all other points. In this case there is just one sender, but the information is sent to all connected receivers. Broadcast transmission is supported on most LANs, and may be used to send the same message to all computers on the LAN. Network layer protocols also support a form of broadcast that allows the same packet to be sent to every system in a logical network.

2.6.3. Multicast Routing

Multicast is used to describe communication where a piece of information is sent from one or more points to a set of other points. In this case there is may be one or more senders, and the information is distributed to a set of receivers (there may be no receivers, or any other number of receivers). Multicasting is the networking technique of delivering the same packet simultaneously to a group of clients. IP multicast provides dynamic many-to-many connectivity between a set of senders and a group of receivers. The format of an IP multicast packet is identical to that of unicast packets and is distinguished only by the use of a special class of destination address which denotes a specific multicast group. Since TCP supports only the unicast mode, multicast applications must use the UDP transport protocol [37].

2.7. Routing Decision

QoS routing schemes use collected global state information to make decisions of selecting a feasible path that satisfies flow requirements by computation. These decision processes can be classified into three major classes: source routing, distributed routing and hierarchical routing[38].

2.7.1. Source Routing

Source routing approach computes and makes decisions on feasible paths at the source node. The complete global state information of the network such as network topology and the state of each link in the network is collected and maintained in each node. Source routing sends a setup message from the source node along the computed path to inform each intermediate node about a connection request requirement until the message reaches the destination node. But if it does not find a feasible path, the source node may reject the connection or negotiate for fewer requirements. Among network nodes, global state information is used by either a link state protocol [6] or a distance vector protocol[39].

Source routing has advantages of being simple, flexible, loop free and easy to implement, as the feasible paths are computed in a centralized fashion for each individual connection. Various algorithms can be used in the source node as the paths are computed locally. But source routing needs a very frequent exchange of complete information since it greatly depends on the precision of and the maintenance of the global state information [1][40], and this results in very high computational overhead. Many QoS routing algorithms are based on this approach although it has the problems above[38][41][42]. The different QoS schemes, such as partitioning of large networks to

reduce computational overhead [43], end to end QoS requirements partitioning to reduce routing cost [44], and traffic engineering [45] are often applied.

2.7.2. Distributed Routing

Distributed routing computes feasible paths distributively among the intermediate nodes between source and destination nodes. In most cases each node is required to maintain global state information by distributed routing algorithms [46][47] which make decisions on a hop-by-hop basis. At each node, a routing table that stores the next hops for all destinations is computed periodically. This makes the communication overhead unusually high for large networks. However, distributed routing has less setup time and is more scalable compared to source routing.

Flooding-based [48][49] QoS routing is another approach for distributed routing. It does not require link state information and complex path computation since it has the ability to search and select the best path based on a number of flooded control messages from the source node. When the network state is imprecise, distributed routing may run into the problem of routing loops.

2.7.3. Hierarchical Routing

Hierarchical routing is used to solve the scalability problem in large networks [50][51][52]. Hierarchical routing clusters nodes into groups to form a logical node. To create a hierarchy in the form of a multi level topology, the logical nodes are further clustered into higher level logical nodes. Hierarchical routing needs each node to maintain aggregated network state information about the other clusters and detailed

state information about nodes in its own cluster. The source routing schemes are used to compute feasible paths as connection requests arrive in hierarchical routing. Distributed routing schemes are used to spread the distribution of path computation over many nodes. Hierarchical routing has advantages and performs well with large scale networks, since the size of the aggregated information in hierarchical algorithms is logarithmic to the size of the whole state information, so the computational effort and the exchange of network state overhead is reduced [53]. But when the number of aggregated levels increases, hierarchical routing has a problem of imprecise state information produced by the aggregation [54].

2.8. Summary

This chapter has described the main concepts of QoS routing. It is clear that the main drawback of global QoS routing schemes is their inability to scale well to large networks. That is to say, when the number of metwork nodes becomes very large then the communication overhead involved in handling the global link state and keeping this up-to-date can become prohibitive. Heirachical routing can be used to reduce the size of the link state by aggregating states, but the aggregation results in imprecise information which can cause errors. In the next chapter we discuss related work and, in particular, the concept of localised QoS routing which aims to overcome the scalability problems associated with the global schemes.

Chapter 3

Related Works

3.1. Introduction

As networks modernize and expand with the increasing deployment of high-speed technology, routing protocols that use shortest-path algorithms for single-metric path computation are inadequate for real-time huge-volume data transfer applications which often require guaranteed quality of service (QoS). To support QoS requirements, a routing protocol must supply explicit information on resources available in the network so that applications can make proper resource reservation. So we need a more complex model of the network, taking into account important network parameters such as bandwidth, delay, jitter rate and loss probability. This, however, has raised a number of challenging technical issues for routing protocols. Successful deployment of QoS routing can enable the finding of a feasible path that has adequate resources to satisfy a set of QoS requirements of a connection. Localised QoS routing is a recently proposed approach that tries to circumvent the problems of global QoS routing by having the source nodes to infer the network QoS state based on flow statistics collected locally, and perform flow routing using this localised view of the network QoS state. In this chapter we describe the related works for global QoS routing algorithms and localised QoS routing algorithms with a comparison.

3.2. Classes of QoS Routing Algorithms

For satisfaction of connection requirements, state information collection and path computation are two main tasks of QoS routing. Based on methods for computation of feasible paths and maintaining state information, QoS routing is categorized into global QoS routing algorithms and local QoS routing algorithms. Different QoS routing algorithms can be reviewed in [21][55][56]. Complexity, optimality and scalability are main problems of QoS routing algorithms [11].

Shortest path algorithms find the shortest path between a given source and destination according to some specific criteria so that the cost of links used on the path is kept to a minimum. Dijkstra's algorithm[57] and the Bellman-Ford algorithm[58] are two well-known shortest path algorithms. Using Dijkstra's algorithm, the shortest path from a given source to all destinations in the network can be computed. Instead of Dijkstra's algorithm, which is not valid for negative weight links in a network, the Bellman-Ford algorithm requires the source node to know the cost of the shortest path to all nodes before the destination. So the Bellman-Ford algorithm can be used as a distributed algorithm such as the RIP[39].

3.3. Global Routing Algorithms

For collection of knowledge about a global view of network QoS state information, QoS routing algorithms need exchange of link state information among the network nodes. A source node finds a feasible path to route a flow to the destination node according to the collected information. Various sorts of global routing algorithms use different methods to select paths and exchange global state information, which are introduced as follows.

3.3.1. The QOSPF algorithm

Based on the existing OSPF protocol, the QOSPF algorithm has been developed to improve performance with minimum intrusion. The QOSPF algorithm computes hop count while performing the path selection algorithm, but the bandwidth is advertised to nodes in the network. In the network, each node requires to maintain the information database of network topology and the bandwidth link state. The QOSPF algorithm selects a feasible path by different computation methods of hop count and bandwidth [41].

3.3.2. The Shortest Distance Path Algorithm (SDP)

The shortest distance path algorithm (SDP) is also named bandwidth-inversion shortest path (BSP) which defines a link's distance as the inverse of the available bandwidth of that link. The SDP chooses the path with the shortest link's distance. The distance function is equal to the sum of distances over all the links along the path, that is: $disp(p) = \sum_{i \in p} \frac{1}{w(i)}$, where $w(i)$ is the available bandwidth of link i. The SDP algorithm prefers the least loaded paths and takes into consideration hop counts [59]. The modified distance function can solve the shortest cost, that is $disp(p,n) = \sum_{i \in p} \frac{1}{w(i)^n}$, where $w(i)$ is the available bandwidth of link i, n is used to change the range between the shortest path $(n = 0)$ and widest path $(n \to \infty)$ [59][60].

3.3.3. Enhanced Bandwidth-Inversion Shortest Path Algorithm (EBSP)

The EBSP algorithm is an enhanced SDP algorithm, which adds a penalty to the function weight of the SDP algorithm. The penalty is increased to prevent the paths from being long when the number of hop counts along the path is increased. The distance function is $disp(p) = \sum_{j=1}^{k} \frac{2^{j-1}}{w(i_j)}$, where $w(i)$ is the available bandwidth of link i [61].

3.3.4. The Shortest Widest Path Algorithm (SWP)

The shortest widest algorithm (SWP) is proposed to select the widest feasible path with maximum available bandwidth. In the case of more than one path with the same width, the shortest path is selected. For finding the most feasible path, Dijkstra's algorithm is applied twice. When it meets more than one path with equal bottleneck, SWP uses the second metric (hop count or delay). The advantage of the SWP algorithm is that it can distribute load efficiently in the network and avoid congestion on short paths because it prefers the widest path[27].

3.3.5. The Widest Shortest Path Algorithm (WSP)

In the widest shortest path algorithm (WSP) [41], the shortest feasible path with minimum hop count among paths that satisfies the bandwidth constraints is selected. If more than one path with the same hop count exists, the path with the maximum available bandwidth is selected. The widest path is selected in the case of more than one

path with the same length. The usage of network resources is minimized by preferring the shortest path to the destination. WSP is used on a pruned topology since links that do not satisfy flow requirements are eliminated[62]. Based on choosing different cost functions of the shortest path, the WSP algorithm has been studied extensively in the literature[63][64][65].

3.4. Localised Routing Algorithms

3.4.1. Proposition of Localised Routing Algorithms

In most source routing algorithms, each source node must have global QoS state information of the network in order to perform routing [55]. This global state is typically updated periodically by a link-state algorithm and maintaining it up-to-date gives rise to several problems such as high communication overhead and route flapping which results from global synchronisation of distributing the network state. Localised QoS routing [2][66] is a relatively recently proposed approach that tries to circumvent these problems by having the source nodes to infer the network QoS state based on flow statistics collected locally, and perform flow routing using this localised view of the network QoS state. Localised QoS routing needs each source node to first determine a set of candidate paths to each possible destination. Candidate path selection is an important factor and various methods exist to do this [67][68]. How to select a suitable path is a key issue in localised QoS routing. The most relevant works are the proportional sticky routing (PSR) [2] and the Credit based routing (CBR) [3] algorithms. In the following we will describe these two algorithms. As a reference, an original localised routing scheme, Dynamic Alternative Routing (DAR)[69], will also be introduced.

3.4.2. The Dynamic Alternative Routing Algorithm (DAR)

The Dynamic Alternative Routing Algorithm (DAR)[69] is an original localised routing scheme which has been used in the Public Switched Telephone Network (PSTN). DAR is a distributed, adaptive routing scheme and is limited to maximum two-link routing and employs trunk reservation. In this scheme, a flow is routed according to the feedback received from the previous accepted or rejected flow. At first the source node tries to route the call along a direct one-link path to the destination. If the call is rejected, a preferred two-link path is then chosen to route the call. When the call cannot be routed along the preferred two-link path, the call is blocked. Then another two-link path is selected from all two link paths as the preferred path.

Figure 3.1 shows a flow diagram for DAR algorithm. In a PSTN, the originating switch i maintains an alternate path k in its cache to each destination switch j. When a new call arrives, DAR first attempts the direct link i - j. If the direct link i - j has no available capacity, the alternate path (i - k , k - j) in its cache is tried. If the call is accepted on this alternate path, the alternate path remains in the cache. But if there is non-availability of capacity on the alternate path, the call is lost. Then the originating node i selects an intermediate node randomly and sticks this in the cache as a new alternate path for the next call to use. So an alternate path remains in the cache only if any calls using this alternate path are successfully connected; if the current alternate route cannot connect a call using this path, a new alternative path is chosen randomly.

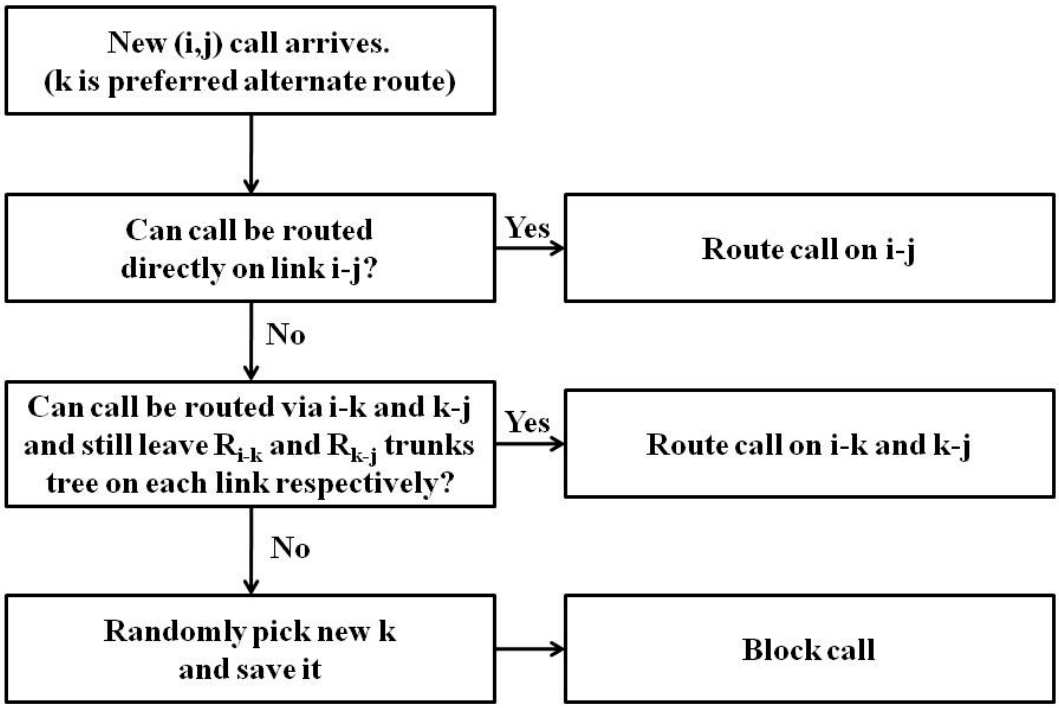


Figure 3.1 Flow diagram for DAR algorithm (taken from[69]).

3.4.3. Proportional Sticky Routing (PSR)

Under the PSR scheme, a predetermined set of candidate paths R to each possible destination is maintained by every node. Based on the route-level statistic of the number of flows blocked, which is the only available QoS state information, a source proportionally distributes the load to a destination among multiple paths by observing the flow blocking probability. PSR considers two types of paths, minhop paths $R^{\min}$ and alternative paths R^{alt}, where $R = R^{\min} \cup R^{alt}$, and prefers minhop paths while using the alternative paths to limit the so-called "knock-on" effect [2][70]. There are two

stages in the PSR scheme: (1) the proportional flow routing, and (2) the computation of flow proportions. Figure 3.2 shows the pseudo-code for PSR.

```
1.   PROCEDURE PSR-ROUTE()
2.     Select an eligible path r from R^elg
3.     Increment flow counter, n_r = n_r + 1
4.     If failed to setup connection along r
5.       Decrement failure counter, f_r = f_r - 1
6.       If failures reached limit, f_r == 0
7.         Remove r from eligible set, R^elg = R^elg - r
8.         If eligible set is empty, R^elg == ∅
9.           Reset eligible set, R^elg = R
10.          For each path r ∈ R
11.            Reset failure counter, f_r = γ_r
12.  END PROCEDURE
```

(a)

```
1.   PROCEDURE PSR-PROPO-COMPU()
2.     For each path r ∈ R
3.       Compute blocking probability, b_r = ηγ_r / n_r
4.       Assign a proportion, α_r = n_r / Σ_{r̂∈R} n_r̂
5.     Set target blocking probability, b* = min_{r∈R^min} b_r
6.     For each alternative path r' ∈ R^alt
7.       If blocking probability high, b_r' >= b*
8.         Decrement failure limit, γ_r' = γ_r' - 1
9.       If blocking probability low, b_r' < ψb*
10.        Increment failure limit, γ_r' = γ_r' + 1
11.  END PROCEDURE
```

(b)

Figure 3.2 Pseudo-code for PSR (a) Proportional routing, (b) Computation of Proportions. (taken from [2]).

In the proportional flow routing stage, incoming flows are routed along paths which are selected from a set of eligible paths R^{elg}. A path r is selected from this set with a frequency determined by a prescribed proportion α_r. A variable γ_r is named as the maximum permissible flow blocking parameter and f_r is defined as the corresponding flow blocking counter. For each minhop path, $\gamma_r = \hat{\gamma}$, where $\hat{\gamma}$ is a configurable system parameter. For each alternative path, the value of $\gamma_{r'}$ is dynamically adjusted between 1 and $\hat{\gamma}$. When what is called a cycle begins, f_r is set to γ_r. f_r is decremented when a

flow routed along path r is blocked, and when f_r reaches zero, path r is considered ineligible. As soon as R^{elg} becomes empty a new cycle is started with $R^{elg} = R$ and $f_r = \gamma_r$. In the computation of flow proportions stage, α_r and b_r are recomputed at the end of each observation period which consists of η cycles, where b_r is the flow blocking probability on path r. It shows that flow blocking rates (α_r b_r) for minhop paths are equal. The minimum blocking probability among the minhop paths b^* is used as the reference to control flow proportions for the alternative paths. The minhop paths are always preferred to alternative paths in routing flows. Some flows are routed along alternative paths to measure their quality [2].

3.4.4. Credit Based Routing (CBR)

As with PSR, every node maintains a predetermined set of candidate paths R to each possible destination in CBR. CBR differentiates between two types of paths, the minhop paths set $R^{\min}$ and alternative paths set R^{alt}, where $R = R^{\min} \cup R^{alt}$. Associated with every path P there is a variable $P.credits$ which stores the accumulated credits gained so far. The pseudo-code for CBR is shown in Figure 3.3.

```
Initialise
  set P.credits = MAX_CREDITS, ∀ P ∈ R
CBR(MAX_CREDITS)
  1. if P.credits = 0 ∀ P ∈ R
  2.    set P.credits = MAX_CREDITS, ∀ P ∈ R
  3. P^min = max{P.credits : P ∈ R^min}.
  4. P^alt = max{P.credits : P ∈ R^alt}.
  5. if(P^min.credits >= Φ × P^alt.credits)
  6.    set P = P^min
  7. else
  8.    set P = P^alt
  9. route flow along path P.
 10.    if flow accepted
 11.    UpdateBlockingProbability(P)
 12.    amount = (1 − P.BlockingProbability)
 13.    P.credits = min
        {P.credits + amount, MAX_CREDITS}
 14. else
 15.    UpdateBlockingProbability(P)
 16.    amount = (P.BlockingProbability)
 17.    P.credits = max{P.credits − amount, 0}
```

Figure 3.3 The pseudo-code for CBR (taken from [3]).

Initially, *P.credits* is set to *MAX _ CREDITS* which is a system parameter representing the maximum attainable credits for each path. When flows arrive, two paths $P^{\min}$ and P^{alt} are compared, which are the paths with maximum credits in $R^{\min}$ and R^{alt} respectively. If $P^{\min}.credits \geq \Phi \times P^{alt}$ CBR routes the flow along $P^{\min}$; otherwise, P^{alt} is chosen. Φ is a system parameter that controls the usage of alternative paths and limits the "knock-on" effect. When the flow is accepted along a selected path

then, $P.credits = \min\{P.credits + amount, MAX\_CREDITS\}$; that is, the credits for that path are updated, where $amount = (1 - P.Blocking\ \Pr obability)$, and so $P.credits$ is incremented to correspond to its success probability. If the flow is rejected, its blocking probability is updated accordingly $P.credits = \max\{P.credits - amount, 0\}$ where $amount = (P.Blocking\ \Pr obability)$, and so $P.credits$ is decremented to correspond to its blocking probability. To continuously monitor flow blocking probabilities, CBR uses a simple moving average with predetermined period to calculate a variable s, which is and estimate of the blocking probability for every path. In a period of M flows requests, s is calculated by the most recent M flows along the path. For example, if s ={1, 1, 0 1, 0} represents the data of the last M =5 flows, where 1 indicates flow acceptance and 0 indicates flow rejection, then the blocking probability is 2/5. Then if another flow is accepted, the oldest element will be deleted from s and replaced by the data from the last flow, i.e. s ={1, 1, 1, 0, 1} and the blocking probability is updated accordingly to 1/5.[3]

Note that with both PSR and CBR their key decision parameters are based on the blocking of flows on each path in the candidate path sets and these are only updated for a specific path when that path is requested and used. These parameters can therefore become stale if the paths are not requested and chosen very often and yet they are used to calculate the flow proportions in PSR and the credits in CBR. This can clearly lead to possible inaccuracies in the routing decisions for both these algorithms. Through extensive simulations, CBR has previously been shown to outperform the PSR algorithm in most situations [3] and for this reason we do not use PSR but use CBR as the localised algorithm of choice with which to compare our proposed algorithms.

3.5. Time Complexity

In order to scale QoS routing algorithms to large networks, the time complexity can give an indication of the dominant factors in the number of steps an algorithm takes to solve a problem. The time complexity of a QoS routing algorithm is related to the number of operations needed to satisfy QoS requirements and their composition rules. Since all computational steps to find a feasible path are carried out in the source, time complexity is a major performance criterion in QoS source algorithms. Table 3.1 show the time complexity of some common QoS routing algorithms[3].

Routing Algorithm	**State-Information**	**Time Complexity**
SDP	Global	$O(N \log N + L)$
SWP		$O(N \log N + L)$
WSP		$O(N \log N + L)$
Dijkstra		$O(N \log N + L)$
PSR	Local	$O(R)$
BR		$O(R)$

Table3.1 Time complexity of QoS routing algorithms

(N **=number of nodes,** L **=number of links,** R **=number of candidate paths)**

3.6. Summary

This chapter has discussed specific QoS routing algorithms that currently exist or have been proposed in the literature. In particular, it is clear that the problems associated with the scalability of global QoS routing schemes can be overcome to some extent by the use of localised QoS routing. This is because localised algorithms do not require to store and update the global link state but infer this latter by the use only of local information such as the state of the links to the nodes that are immediate neighbours and/or the blocking probability of the candidate paths. Such information is either immediately available to a node or can be easily computed. This advantage is also reflected analytically in the superior time complexity of the localised algorithms. For this reason we focus the remaining chapters of the thesis on the development and performance evaluation of new localised routing algorithms and compare these with existing local and global algorithms.

Chapter 4

Simulation Platform

4.1 Introduction

In this chapter, we set up a simulation platform to evaluate the performance of the proposed algorithms. After comparison of simulators, we select OMNeT++ as the most appropriate simulator among existing ones. The simulation we developed is assumed to emulate the real Internet under different scenarios. We discuss different aspects of network graph models and network topologies. Otherwise, some adopted parameters and performance measures for simulating the proposed algorithms are also discussed.

4.2 Discussion of Simulation

In order to research the behaviour of a communication system, we have to develop a model which can represent this system. Analytical modelling and simulation are two approaches for development of the models. The analytical modelling approach includes developing a solution for the concerned system. This solution is developed by using mathematical equations which usually incorporate a number of numerical parameters which are called performance measures of a system[71]. On occasion it is quite difficult to develop such models due to the nature of the modelled system such as the type of routing algorithms. The simulation modelling approach includes developing a computer

program to represent a communication system. The simulation approach is widely used in many areas including communication networks, manufacturing, business, bioscience, military and health[72]. The advantage of simulation is that a system can be studied under different conditions without the need to build a real system. In network simulation, a computer program is developed to mimic the behaviour of a computer network. The requirement of computer networks simulation tools is obvious due to the growing complexity of computer network components. Researchers can use powerful tools of computer network simulation to study and understand these components and predict their behaviour.

There are two tasks in simulation of the performance of quality of service routing algorithms. The first task is the simulation of the quality of service routing algorithm based on the assumptions about the computer network and parameter settings. The second task is to produce critical comments on the first task results. The critical comments should show the performance of the simulated quality of service routing algorithm [73]. The consistency and legitimacy of the second task is dependent on the first one. An assumption and scenarios of the simulation environment parameters which are close to those of a practical computer network is most preferred. In some situations, this is difficult since the network is complex and dynamic. Simulating QoS routing algorithms in real computer networks such as the Internet environment requires complex analytical models of queuing delay at the packet level in the form of arrival and service processes [74], existence of QoS and best-effort network traffic with various connection duration profiles [75], synchronous runs of multiple scheduling policies[76] and congestion control mechanisms[77][78] and also implementation of the proposed QoS architectures[79][80][81][82][83]. Because quality of service routing is a network layer entity, quality of service routing algorithms are unaware of many of these dynamic

micro level conditions. The input to the quality of service routing algorithms would simply be the network state information either global or local and the graph structure which represents the underlying network.

4.3 Description of Simulators

Discrete-event simulation complements analytical (mathematical) tools and experimental methods for performance analysis and estimation of communication networks. A well designed simulation model can provide an extremely important insight into the structural weaknesses of a computer network or a communication protocol. Because of their versatility, in addition to modelling communication networks, discrete event simulation methods are indeed widely used in a wide number of areas such as computer system performance analysis, manufacturing processes, database systems etc. and there are a large number of discrete-event simulation tools available.

4.3.1 NS2

NS2 is an object oriented simulator, written in C++, with an OTcl interpreter as a frontend. The simulator supports a class hierarchy in C++ (also called the compiled hierarchy in this document), and a similar class hierarchy within the OTcl interpreter (also called the interpreted hierarchy in this document). The two hierarchies are closely related to each other; from the user's perspective, there is a one-to-one correspondence between a class in the interpreted hierarchy and one in the compiled hierarchy. The root of this hierarchy is the class TclObject. Users create new simulator objects through the interpreter; these objects are instantiated within the interpreter, and are closely mirrored by a corresponding object in the compiled hierarchy. The interpreted class hierarchy is

automatically established through methods defined in the class TclClass. User instantiated objects are mirrored through methods defined in the class TclObject. There are other hierarchies in the C++ code and OTcl scripts; these other hierarchies are not mirrored in the manner of TclObject.[84]

4.3.2 OPNET

Like NS2, OPNET is also a popular network simulator, targeting a wider range of networks and protocols. NS2, derived from REAL, is an open source network simulator. NS2 is widely used for network research in academia. NS2 is also free. However, NS2 is more difficult to learn and lacks a user interface. It requires the users to learn and use non-standard scripting interfaces such as tcl. It takes a significant amount time to get familiar with NS2. As a network simulator, OPNET is better than NS2 for the following reasons:

• OPNET is much easier to use than NS2. It provides a very convenient Graphical User Interface (GUI) and is very easy to learn.

• OPNET can be used to model the entire network, including its routers, switches, protocols, servers, and the individual applications they support. A large range of communication systems from a single LAN to global inter-networks can be supported.

• OPNET software (with model source code) is available for free to the academic research and teaching community.

• The OPNET's discrete event engine for network simulations is the fastest and most scalable commercially available solution. It usually takes just a few minutes to complete simulations of most lab experiments.

• OPNET has a large user community. OPNET software is used by major fortune-500 companies [85], service providers, and government organizations worldwide. Students who have experiences with OPNET simulator are likely to have better future employment opportunities in the networking industry.[85]

4.3.3 OMNeT++

OMNeT++ is a discrete event-driven simulator that provides a rich set of functions and tools for simulating the elements of a communication network, such as nodes, links and packets. OMNeT++ stands for Objective Modular Network Testbed in C++. OMNeT++ is a very well designed, modular, widely-used system, source code is available and free for teaching and research purposes. The name itself stands for Objective Modular Network Testbed in C++.

The main advantages of OMNeT++ are:

- We do not have to learn new programming languages, so we can write all our code in C++.
- We are offered the use of a graphical user interface,
- The whole simulation is portable between Unix-based systems, Windows (9x and NTs) and DOS.
- Many different structures can be simulated without modifying the source code and rebuilding the system, using the same parameters.
- We do not have to write the C++ code to build the modules used in the simulation (objects, gates): The very easy NED (Network Description Language) compiler does it.
- The GUI offers many possibilities to follow and debug the simulation flow.
- We are offered to use parameter-watching functions and plot diagrams of them.

- We can use many pre-defined classes (topology support which contains graph algorithms, finite deterministic state machine support, etc.), and many other implemented modules (TCP/IP, routing models) under OMNeT++, their number grows dynamically.
- OMNeT++ has clear, well-defined, documented and hierarchically nested modules, which are available in source code, so we can debug and/or extend them.
- We can run the simulation on many machines at the same time using the PVM support.

Among the simulator above, we select OMNeT++ to build our simulation models and do our experiments.

The basic workflow using the OMNeT++ is the following:

- Give the network structure to the framework (e.g., using the NED language, may make the connections parameter-based),
- implement the modules which stand at the lowest hierarchy level (in C++),
- compile and link them,
- afterwards let the framework call our modules and handle all the message queues, etc.
- In the end we can evaluate results generated by the built-in parameter-watching functions, and the figures graphically produced by GNU plot.

Since the connections and module types can be parameter-dependent (e.g. this means, we can implement a terminal module more than once and there is a condition of some parameters which one shall be executed and how they shall be connected to each other) we can rerun the simulation and evaluate results using different “versions” of modules and connections. We are also able to make it in a loop if we would test a large amount of alternatives.[86][87]

4.4 Graph Model

For simulating a quality of service routing algorithm, it is important to choose a graph model that is close to a real computer network topology. If the evaluation is carried out on an inappropriate topology, the performance of a routing algorithm may give imprecise results. Because of its rapid evaluation, it is difficult to model a structure for the Internet [88]. For example, choosing a network topology with the size, node degrees and path lengths between pairs of nodes that are huge may turn into an algorithm that is NP-complete [89]. Networks can be categorised according to topological properties and their characteristics can be summarized by the degree of a node, clustering of nodes and shortest-path length between any two nodes [90]. There are a lot of existing models which can be used to generate topologies that need some or all the above characteristics. However the models which are relevant and commonly used in this thesis will be described.

4.4.1. Random Topology

The random graph models were studied to create a more realistic network by many early efforts [91]. In the random graph, links are added with probability of a function of the Euclidean distance between any pair of nodes. There are different graph models which produce different distributions of probability on graphs [92]. The two most important random graphs are the Waxman [93] and the Doar-Leslie [94] graph models which are discussed as follows.

4.4.1.1 Waxman Graph

In a Waxman graph model, nodes and links are created by the following methods: the nodes of the network are uniformly distributed in a 2-dimensional Cartesian coordinate space, and links are added according to probabilities that depend on the Euclidean distance between the nodes. Between a pair (m,n) of nodes, the probability to add a link is:

$P(m,n) = \beta e^{-d(m,n)/\alpha L}$ [93]

Where $\beta > 0$, $\alpha \leq 1$, $d(m,n)$ is the distance from m to n, and L is the maximum distance between any two nodes in the graph.

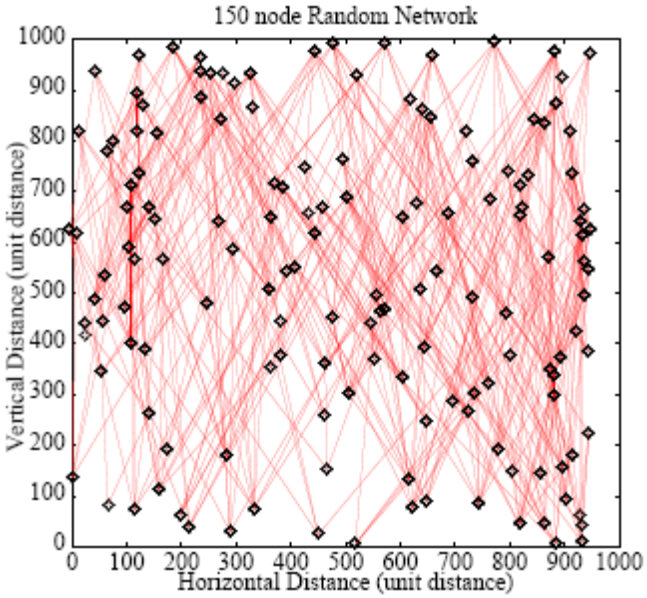


Figure 4.1 A type Waxman graph (adapted from [95])

Figure 4.1 shows a type of Waxman graph which $\alpha = 0.25$, $\beta = 0.3$. A large value of α increases the number of connections to nodes further away, whilst a large value of β increases the number of edges from each node. A Waxman graph has a problem that the

nodes require impractical node degrees when the number of nodes increases, although they are widely used and simple to implement.

4.4.1.2 Doar-Leslie Graph

For solving the problem of a Waxman graph, Doar and Leslie proposed a modified model so as to limit the average node degree increase. A scaling factor (KD/N) is added to Waxman's link probability equation to stabilize the average node degree, where K is the scale factor and D is the mean degree of the node.

The modified equation is:

$P(m,n) = (KD/N)\beta e^{-d(m,n)/\alpha L}$.[94]

The Doar-Leslie model is used to generate random network topologies for simulation with certain topological characteristics in this thesis. An example of a 150-node random topology generated using the Doar-Leslie model is shown as in Figure 4.2.

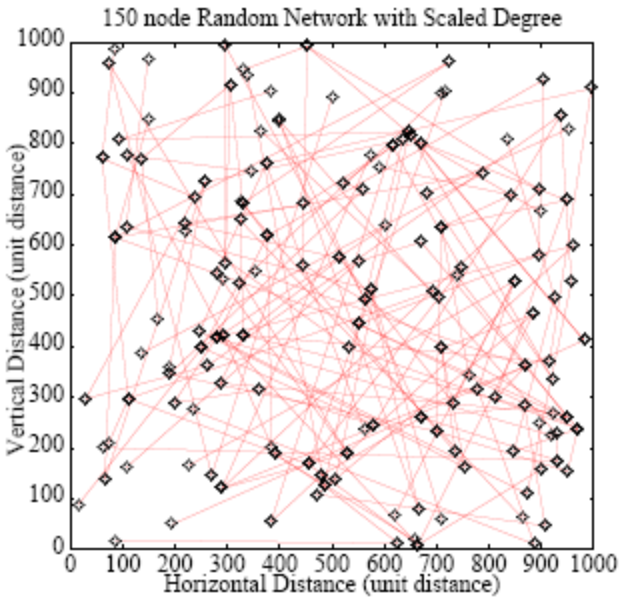


Figure 4.2 Doar-Leslie graph (adapted from [95])

4.4.2. Regular Topology

In a regular graph, each node has the same number of neighbours; if k represents the node degree then a k-regular graph means a regular graph with node degree k. Regular graphs are generally sublimate topologies such as lattice, torus, star and ring topologies. They are regularly used to test some features of Internet performance. In this thesis the torus topology is made use of in our simulation. A 4-node torus graph can be seen in Figure 4.3.

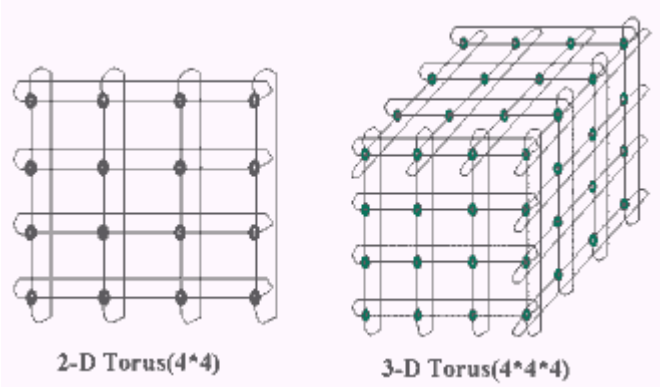


Figure 4.3 2-D and 3-D Torus topologies with 4 nodes

4.4.3. ISP Topology

Internet Service Providers (ISP) have their own network topological structure to optimize the network traffic performance. An ISP topology can be considered as a single autonomous system (AS) domain. It consists of hundreds of interconnected routers and points of presence (POPs)[96][97]. The modified ANSNET [98] is the most widely used ISP topology for performance evaluation of quality of service routing

algorithms [99][100], which is shown in Figure 4.4. The ISP topology has been extensively used in the simulation of routing algorithms [3][101][102]. It is also used in this thesis to evaluate the performance of the proposed localised quality of service algorithms.

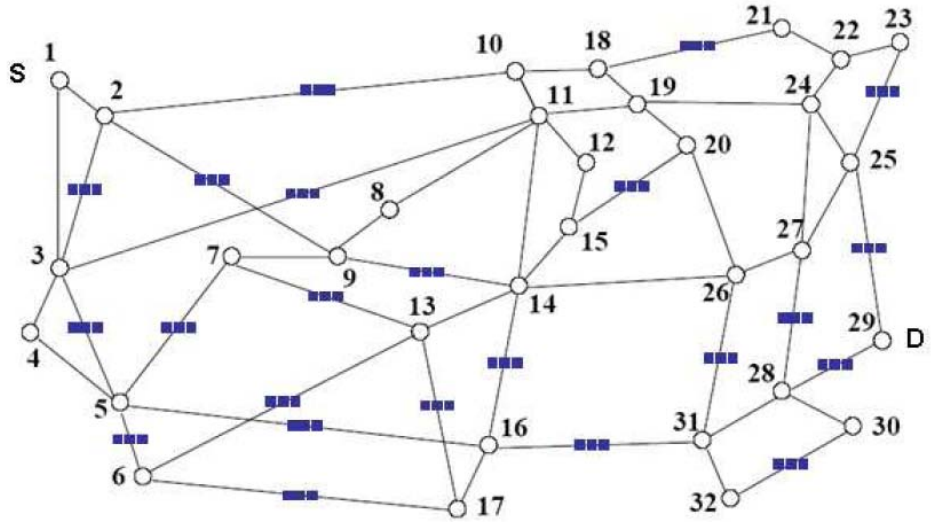


Figure 4.4 ISP topology

4.5 Simulator Design

Designing simulators is an important and necessary step in evaluating the performance of the proposed algorithms. In this section we select a simulator and design its structure.

4.5.1 Simulator Selection

Simulating localised schemes require a simulator which is capable of modelling a relatively large number of nodes. As in our introduction in 4.3, although the NS2 and

OPNET simulator packages are well suited for packet switched networks and because of their focus on low lever modelling such as packet-level, they are mainly used for small scale simulations. Moreover, the outstanding number of simulation events that grow linearly with the number of packets can lead to performance bottlenecks when managing a sorted event list of millions of events. For these reasons, the simulator should be designed from the beginning with scalability in mind. We developed our simulator on top of the OMNeT++. OMNeT++ (Objective Modular Network) is an object-oriented, modular discrete event simulator with an embeddable simulation kernel and GUI support. An OMNeT++ simulation is built on C++ foundations and built out on hierarchically nested modules. Modules are programmed in C++ and use messages as means of communication with each other. A node maintains an arbitrary amount of gates that are used to send messages through links to other nodes. A network topology that contains gates, links and modules, is defined in the Network Description (NED) language [86][87].

4.5.2 Simulator Structure

A functional diagram of a network simulator is illustrated in Figure 4.5. Each functional block performs a particular function in the simulation which is described below.

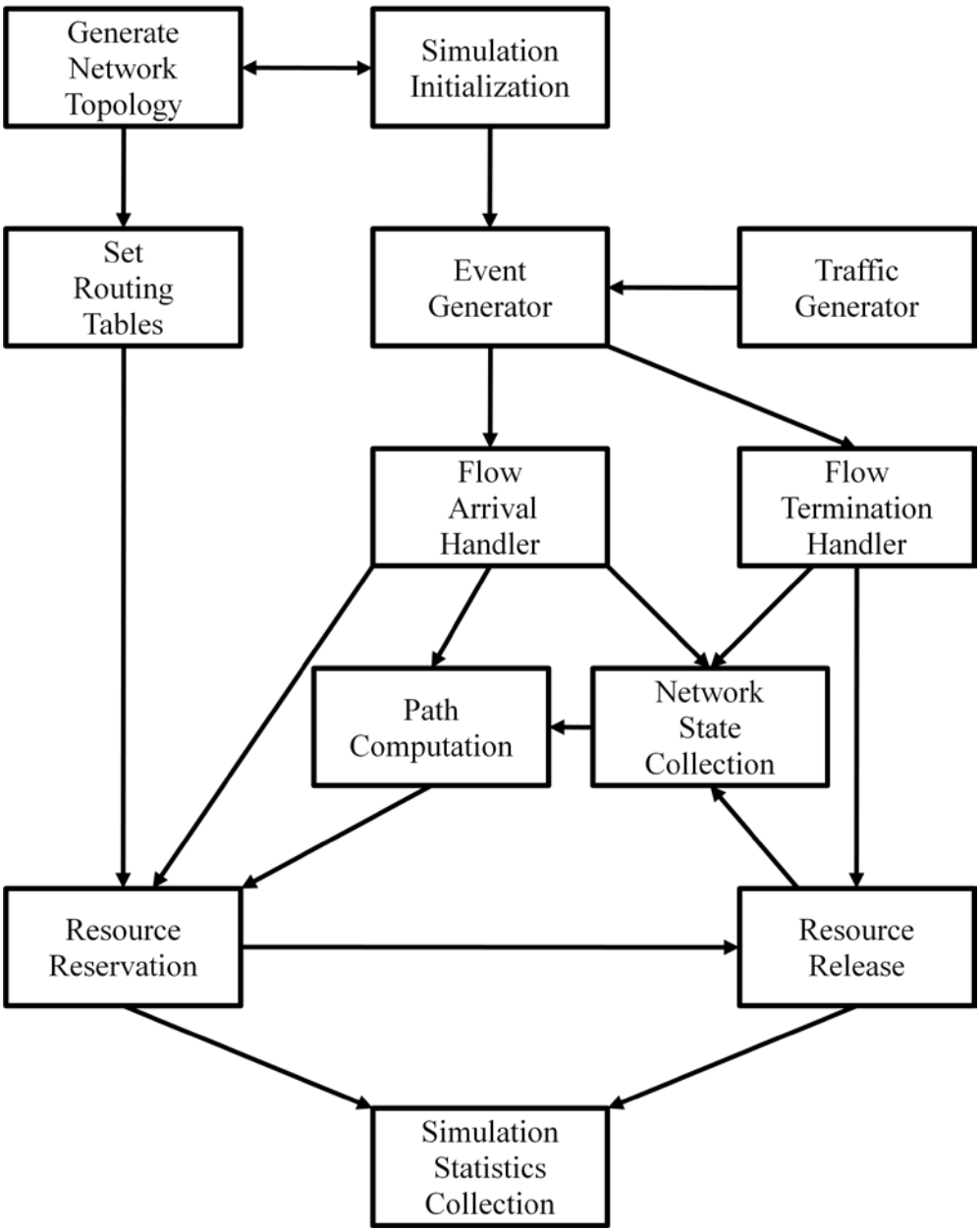


Figure 4.5 Functional diagram of a network simulator

Simulation Initialization

This model initializes the variables used in the entire simulation process. It triggers a "Generate network topology" model and then gets this network topology information.

The initial values are assigned to the topology such as link capacities and link delays and other simulation parameter values.

Generate Network Topology

In this block a random topology is generated, or the regular topology is read from the NED file. Correlative network topology information will be used by other models. The random topology is generated by the Brite generator which is a parametrized topology generation tool, and can be used to flexibly control various parameters (such as connectivity and growth models) and study various properties of generated topologies (such power laws, average path length, etc)[103]. The generation is a four-step process: (1) placing the nodes in the plane, (2) interconnecting the nodes, (3) assigning attributes to topological components (delay and bandwidth for links, AS id for router nodes, etc.) and (4) outputting the topology to a specific format. [103]

Set Routing Tables

In this module routing tables for all nodes in global algorithms and the set of candidate path for each pair of nodes in the localised algorithms are constructed. As soon as the network topology is generated, this model is started up. These routing tables will be used in the resource reservation module to determine the best feasible path to route a flow.

Traffic Generation

This module is responsible for specifying the characteristics of the traffic in the network. When a new connection request arrives to a source node, the traffic generator provides the connection request with a random destination node and the flow duration. The

arrival of the connection requests can be modelled as a Poisson stream or bursty stream with different shaper. The connection request requirements such as the requested bandwidth and delay constraints are also specified in this model.

Event Generation

The event generator model performs two main events according to different instances when a flow (connection request) arrives. If the flow arrives at the node which is the destination of the flow, the flow termination handler model is triggered; otherwise the event generator model activates the flow arrival handler model. When a new connection request arrives to a source node, this model invokes the traffic generator and makes use of the production from traffic generator.

Flow Arrival Handler

The flow arrival handler is the main event handler. It is activated by the arrival of a flow and handles each new connection request and its requirements. The connection request is passed to the path computation module to compute the best feasible path or the most appropriate candidate path for that connection; once the path is determined, the resource reservation module is invoked to signal and reserve resources for the flow. The correlative information is collected by the network state collection model to update the global or localised network state.

Flow Termination Handler

This module releases the reserved resource to terminate the flow by invoking the resource release module, and passes correlative information to the network state collection model to update the global or localised network state.

Network State Collection

This model is responsible for collecting and updating network state for localised or global algorithms. In localised algorithms, the model is triggered after each event finishes in flow arrival handler module and flow termination handler module, or by the resource release module. A source node uses locally collected flows and network statistics generated from itself such as flow blocking probabilities, residual bandwidths and mean delays and updates this information based on this local information. In global algorithms, link state update information is periodically exchanged among network nodes to obtain a global view of the network QoS state. The QoS state of a link may represent the available bandwidth or delay since the last update. Based on this updated localised or global network state, the path computation module determines a feasible path or the most appropriate candidate path for a connection request.

Path Computation

The path computation module implements various routing policies for localised and global routing algorithms. When a connection request arrives, the path computation module attempts to find the best feasible path or the most appropriate candidate path using data from the network state collection module, and passes the path to the resource reservation module.

Resource Reservation

The resource reservation module is implemented for resource reservation, admission control and signalling policy. It is triggered by the flow arrival handler and interacts with the path computation module and the set routing tables module. The signalling

process is initiated hop-by-hop from the source node to reserve network resources for the connection arrival.

The resource reservation module gets the determined feasible path or the most appropriate candidate path p from the path computation module, and reads a specific gate g from the routing tables, and then sends the flow to the gate g to route the flow along the path p. We discuss different cases for bandwidth and delay based algorithms below:

- In the case of bandwidth based algorithms:

  We assume that a flow requests QoS bandwidth b, and each link l in the network has the available residual bandwidth $bw(l)$. When the signaling message passes along the selected path p, each node performs an admission check to examine the outgoing link to make sure it has sufficient residual bandwidth. If the available residual bandwidth $bw(l)$ over the outgoing link is equal to or more than the requested QoS bandwidth b, the node reserves the bandwidth b for the flow so that $bw(l) = bw(l) - b$ and the message is passed to next node in the path p. This module accepts the flow if all links along the selected path p have enough residual bandwidth; otherwise a failure message is transmitted back to the source node releasing the reserved bandwidths so that $bw(l) = bw(l) + b$ and the flow is rejected.

- In the case of delay based algorithms:

  We assume that the flow requested QoS mean delay has value D_r, its cumulative mean delay is D, and each node n in the network has delay (sum of mean delay of flows on the node) $d(n)$. When the signaling message passes through the selected path p, each node carries out an admission check over the outgoing link adding its

delay D_r to next node $d(n)$ to make sure that the flow mean delay D is not more than the requested delay constraint D_r. If the mean delay D over the outgoing link is less than the requested delay constraint D_r the message is passed to next node in the path. This module accepts the flow if the delay along the selected path p is less or equal to the delay constraint and also the delay constraint of any existing flow is not exceeded. Otherwise, a failure message is transmitted back to the source node releasing the reserved resources and the flow is rejected.

This model interacts with the resource release module once the flow duration of a flow has elapsed to release the resources reserved by that flow. This module does not reroute the flow to an alternative path when a failure setup message occurs and as a result the flow is rejected. Although rerouting of flows may decrease the probability of blocking, it would also increase the signalling overhead. It passes correlative information to the simulation statistics collection module.

Resource Release

This module is triggered by the flow termination handler module and resource reservation module to release the reserved resource such as bandwidth and delay. It also passes correlative information to the network state collection model to update the global or localised network state, and to the simulation statistics collection module.

Simulation Statistics Collection

The simulation statistics collection module monitors a mass of statistics during the simulation runs. It is invoked by the resource reservation module and the resource release module in order to collect different aspects of the performance metrics. It computes and provides statistical results as required.

4.5.3 Building and Running Simulations

To build and run simulations based on the simulator structure, we make use of OMNeT++ with C++ to implement routing algorithms.

An OMNeT++ model consists of the following three parts: (1) NED language topology description(s) which describe the module structure with parameters, gates etc. (2) Message definitions. We can define various message types and add data fields to them. OMNeT++ will translate message definitions into full-fledged C++ classes. (3)Simple modules sources. They are C++ files.

The simulation system provides the following two components: (1) Simulation kernel. This contains the code that manages the simulation and the simulation class library. It is written in C++, compiled and put together to form a library. (2) User interfaces. OMNeT++ user interfaces are used in simulation execution, to facilitate debugging, demonstration, or batch execution of simulations. There are several user interfaces, written in C++, compiled and put together into libraries.

Simulation programs are built from the above components. First, the NED files are compiled into C++ source code, using the NEDC compiler which is part of OMNeT++. Then all C++ sources are compiled and linked with the simulation kernel and a user interface to form a simulation executable. [86]

4.5.3.1 Describing the Module Structure of Topologies

The topology of a model is specified using the NED language. The NED language facilitates the modular description of a network. This means that a network description

may consist of a number of component descriptions (channels, simple/compound module types). The channels, simple modules and compound modules of one network description can be reused in another network description. Files containing network descriptions generally have a .ned suffix. NED files can be loaded dynamically into simulation programs, or translated into C++ by the NED compiler and linked into the simulation executable. In the simulator structure, a NED file is a part of the module of "Generate Network Topology".

An example of the NED language topology description for the torus is as below:

```
simple CBR_Node
   parameters:
      address : numeric;
   gates:
      in: in[];
      out: out[];
endsimple

module CBR
   parameters:
      height : numeric const,
      width : numeric const;
   submodules:
      node: CBR_Node[height*width];
         parameters:
            address = index;
```

```
        display: "p=,,m,$width;i=misc/node_s";
    connections nocheck:
        for i=0..height-1, j=0..width-1 do
            node[i*width+j].out++ --> node[(i+1)*width+j].in++ if i!=height-1;
            node[i*width+j].in++ <-- node[(i+1)*width+j].out++ if i!=height-1;
            node[i*width+j].out++ --> node[i*width+j+1].in++ if j!=width-1;
            node[i*width+j].in++ <-- node[i*width+j+1].out++ if j!=width-1;
            node[i*width+j].out++ --> node[(i-height+1)*width+j].in++ if i==height-1;
            node[i*width+j].in++ <-- node[(i-height+1)*width+j].out++ if i==height-1;
            node[i*width+j].out++ --> node[i*width+j-width+1].in++ if j==width-1;
            node[i*width+j].in++ <-- node[i*width+j-width+1].out++ if j==width-1;
        endfor;
endmodule

network cbr : CBR
    parameters:
        height = input(8,"Number of rows"),
        width = input(10,"Number of columns");
endnetwork
```

4.5.3.2 Message Definitions

We can define various message types and add data fields to them. cMessage is a central class in OMNeT++. Objects of cMessage and subclasses may model a number of things: events; messages; packets, frames, cells, bits or signals travelling in a network; entities

travelling in a system and so on. Message definitions are used in the “Traffic Generation” module of the simulator structure.

An example of message definition is as follows:

```
message Packet
{
   fields:
       int source;
       int destination;
       double BandWidth;
       int HopCount=0;
           int pathNode[80];
           int pathGate[80];
           int type;
}
```

4.5.3.3 Simple Modules Sources

In OMNeT++, events occur inside simple modules. Simple modules encapsulate C++ code that generates events and reacts to events, in other words, implements the behaviour of the model. We can create simple module types by subclassing the cSimpleModule class, which is part of the OMNeT++ class library. cSimpleModule, just as cCompoundModule, is derived from a common base class, cModule. These member functions are void initialize(), void handleMessage(cMessage *msg), void activity() and void finish().

In the initialization step, OMNeT++ builds the network: it creates the necessary simple and compound modules and connects them according to the NED definitions. It triggers the “Generate network topology” and “Set Routing Tables” modules

The handleMessage() and activity() functions are called during event processing. This means that the user will implement the model’s behavior in these functions. handleMessage() and activity() implement different event processing strategies: for each simple module, the user has to redefine exactly one of these functions. handleMessage() is a method that is called by the simulation kernel when the module receives a message. In our simulation we prefer handleMessage() to activity()because activity() doesn’t scale well. This is a main step of our experiment. Most modules in the simulator structure are run in the handleMessage() function. We complete the functions of generating events, flow arrival or termination handling, network state collection, path computation, resource reservation and resource release.

The finish() functions are called when the simulation terminates successfully. We complete simulation statistics collection during our simulation.[86]

4.6 Performance Metrics

Performance metrics are used to evaluate the performance of the proposed algorithms. In this thesis we consider two types of performance metrics: Blocking Probability and Jain’s Index, where this latter is used to measure load balance.

4.6.1 Blocking Probability

The aim of developing new QoS routing algorithms is to produce schemes which can satisfy the required QoS in the most efficient way. That is, if a path satisfies the required flow QoS, then the path is accepted; if not, it is rejected. So a value which can reflect the degree of the satisfaction must be the best metric for the proposed algorithms. Then blocking probabilities can be used as measures of the efficiency of the QoS routing algorithms used.

Flows will be rejected when at least one of the links along the path from source to destination does not satisfy the requested bandwidth or the requested delay constraint. The flow blocking probability is defined as:

$$\text{flow blocking probability} = \frac{|B|}{|C|},$$

where B is the set of blocked flows and C is the set of the total flows.

In bandwidth based QoS routing, since flows may request different amounts of bandwidth we also use the notion of bandwidth blocking probability which is defined as:

$$\text{bandwidth blocking probability} = \frac{\sum_{f \in B} bandwidth(f)}{\sum_{f \in C} bandwidth(f)},$$

where B is the set of blocked flows and C is the set of the total flows and $bandwidth(f)$ is the requested bandwidth for path f.

4.6.2 Fairness

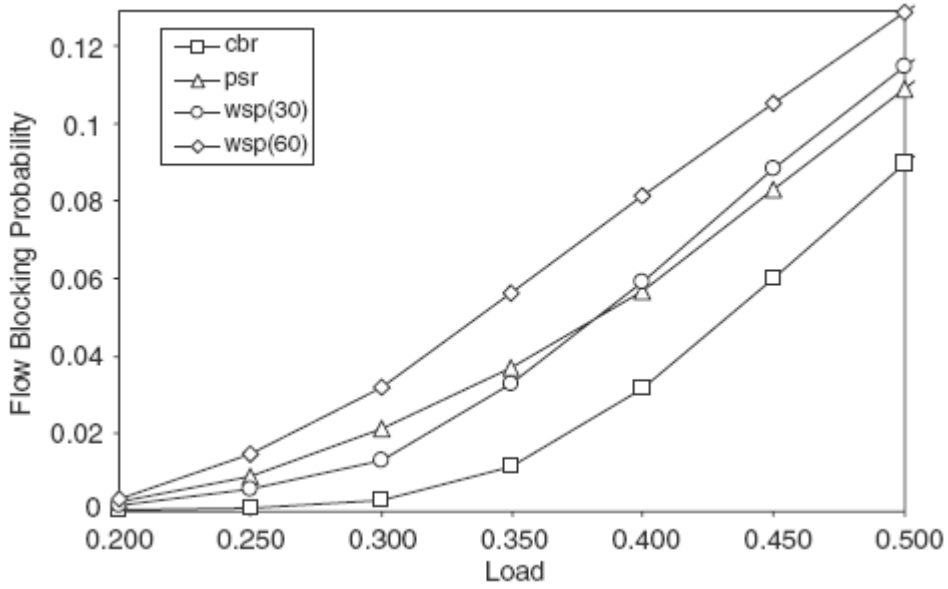
Fairness is used to reflect load balancing which is another important factor to measure the performance of QoS routing algorithms. The aim of the measurement is to use the

network resources in a more efficient manner in order to reduce the risk of network traffic congestion. A load-balanced network has less delay and packet loss than one with an imbalanced load. It is important to design QoS routing algorithms to fairly distribute the load among the links in network topologies.

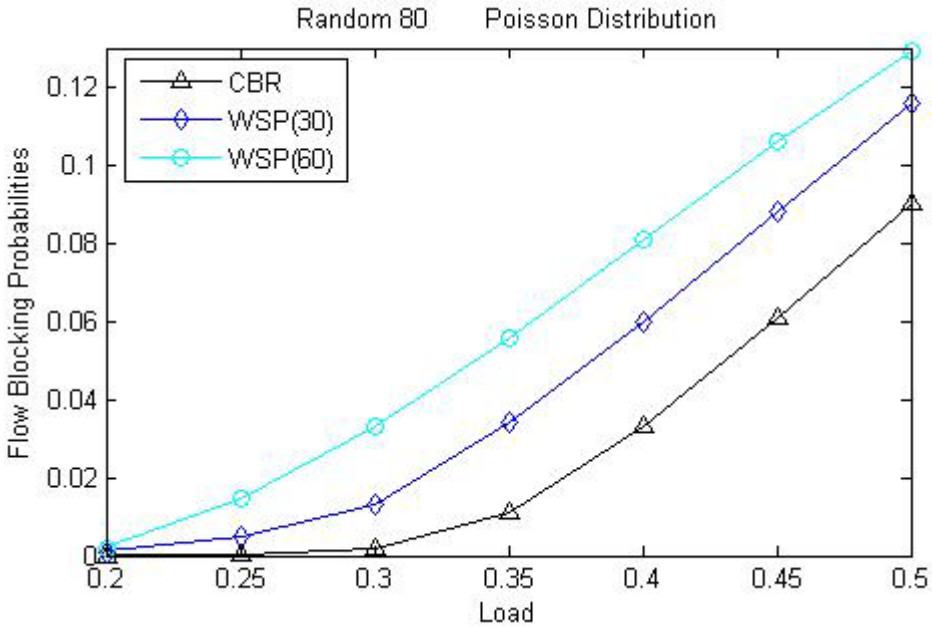
There are several well known definitions of fairness. Jain's Fairness Index (JFI) [104][105] is widely used for assessing system-wide fairness, and is defined as follows, $J = \frac{(\sum_{n=1}^{N} r_n)^2}{N\sum_{n=1}^{N} r_n^2}$, where, $\frac{1}{N} \le J \le 1$, N can, for example, be the number of flows or number of links, r_n is the value of the resource attribute being assessed for flow n, e.g. r_n can represent bandwidth carried by each link of a network. Then $J = 1$ corresponds to fairness across all links and $J = \frac{1}{N}$ indicates no fairness. Thus, if $N \to \infty$ then $J \to 0$.[106]

4.7. Simulator Validation

Validation is defined as the process of ensuring that a simulation represents reality at a given confidence level. Validation confirms simulation as a reasonable representation of the actual system[107]. Simulator validation is a method to demonstrate that the results obtained are correct. We implement the validation process through the simulations in OMNeT++ to verify the correctness of the localised and global QoS routing algorithms.



a Original results taken from [3]



b Verified results

Figure 4.6 Simulator validation results

For the credit based routing algorithm (CBR) and WSP algorithm, we use the same simulation configuration and parameters as described in [3] to repeat the simulations using our simulator. The results are shown in Figure 4.6, and this indicates that the

results are very close to the results reported in [3], which gives confidence that the simulator is performing as it should.

4.8. Summary

In this chapter we have discussed the simulation of the performance of quality of service routing algorithms. After describing and comparing some main discrete-event simulation tools, we selected OMNeT++ as our simulator. We developed a graph model and the simulation model to evaluate the performance of the proposed algorithms in later chapters. We also described the simulator design and validation. Different types of network topologies and the performance metrics such as blocking probability and Jain's Fairness Index used in the performance evaluation were also described. The simulator validation results demonstrated that our simulation platform appears reasonable in that it gave results consistent with those obtained independently by other researchers.

Chapter 5

Localised Bandwidth Based QoS Routing

5.1. Introduction

QoS routing is the key technology to ensure Quality of Service in networks. The aim is to find out an optimal path to meet flow QoS requirements such as specific bandwidths or delays. Most of the QoS routing algorithms[1][108][109][110][111][112] [113][114] require global QoS state information of the network to perform routing. Frequent QoS state updates have to be used to exchange link QoS state information. This brings about several problems such as high communication overhead and route flapping which result from global synchronisation. Although some remedial solutions has been proposed [111][114], the essential problem has still not been completely eliminated.

Localised QoS routing[2] has been proposed to solve some of these problems. Instead of global QoS state information, source nodes infer the network QoS state based on flow blocking statistics collected locally, and perform flow routing using this localised view of the network QoS state. Localised QoS routing has an advantage over other approaches because communication overhead is minimized and the processing and memory at core routers is reduced. Various localised QoS routing algorithms have been put forward such as the proportional sticky routing algorithm (PSR)[2] and the credit based routing algorithm (CBR)[3].

Although these show good performance against other QoS routing schemes, PSR and CBR do not directly reflect the network QoS state fully since they only reflect it indirectly by the addition or subtraction of credits or computation of flow proportions. It is therefore likely that algorithms which use the required QoS metric like bandwidth to directly reflect the quality of a path may result in a better performance.

In this chapter we propose two schemes - the localised Score Based QoS Routing (SBR) and the localised Bandwidth Based QoS Routing (BBR) – which are simple localised QoS routing algorithms that rely on residual bandwidth on the path in order to make routing decisions. We compare their performance with other schemes in terms of flow and bandwidth rejection probability under different network loads and topologies.

5.2. Bandwidth for QoS Routing

For reflecting real-time bandwidth state, we will discuss the example shown in Figure 5.1.

Figure 5.1 is a simple network with seven nodes. Node 0 is the source, and Node 5 and Node 6 are destinations. In the source node, there are four optional paths to next nodes: Path 00, Path 01, Path 02 and Path 03. In the middle nodes of Node 1, Node 2, Node 3 and Node 4, there are two paths to the destinations respectively. At Node 0, each path residual bandwidth is B_{00}, B_{01}, B_{02}, and B_{03}. The problems we must solve are: (1) how to select an optimal path for the flow when a flow arrives at the source; (2) how to update QoS state information of the network after the flow is routed along the selected path.

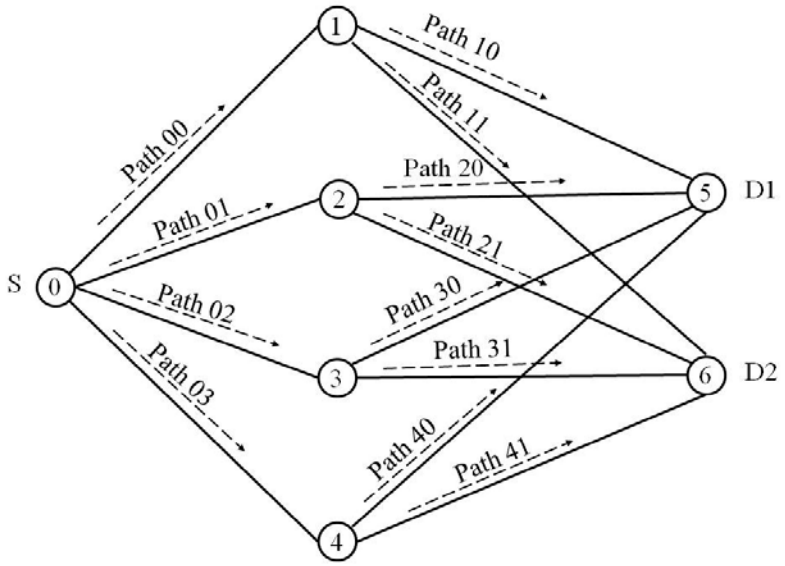


Figure 5.1 An example of Bandwidth for QoS routing

We may consider two schemes to select a path (in this simple explanation of the basic ideas, a path is also a link) described below. In both schemes we assume that every pair of nodes in the network has a predefined set of candidate paths between them. This thesis does not consider the selection of the candidate path sets but a number of methods are available for carrying out this task [2]. Then the basic rationale of both schemes is that we can set up a path from source to destination one hop at a time. Upon arriving at the next node according to some selection criteria, then there exists a set of candidate paths from this new node to the destination. We can thus proceed on the same basis to the next node, and so on until the destination is reached. In this way it is seen that the path set up procedure is distributed rather than based entirely at the source node, as in the existing localised QoS routing algorithms.

5.2.1. Scheme 1: Selecting a Path Using a Random Number Generator

It is logical that the path with maximum residual bandwidth should have the highest probability to be selected for the arrival flow. Then we consider an algorithm to select a path, which uses a variable S_i which is named the score of Path i. S_i is defined as:

$S_i = \frac{B_i}{\sum_{i=0}^{k} B_i}$, where $k =$ number of candidate paths.

In source Node 0, when a flow arrives, the score of Path 00 is $S_{00} = \frac{B_{00}}{B_{00} + B_{01} + B_{02} + B_{03}}$, the score of Path 01 is $S_{01} = \frac{B_{01}}{B_{00} + B_{01} + B_{02} + B_{03}}$, the score of Path 02 is $S_{02} = \frac{B_{02}}{B_{00} + B_{01} + B_{02} + B_{03}}$ and the score of Path 03 is $S_{03} = \frac{B_{03}}{B_{00} + B_{01} + B_{02} + B_{03}}$.

Then we select a path by Random Number Generator as in Figure 5.2.

In Figure 5.2, the Random Number Generator RN generates a random number between 0 and 1.0. The procedure of path selection is as follows:

Run RN generator

if $RN \leq S_{00}$ then

  choose path 00

else if $RN \leq S_{00} + S_{01}$ then

  choose path 01

else if $RN \leq S_{00} + S_{01} + S_{02}$ then

  choose path 02

else choose path 03

then the flow is routed along the chosen path to the next node.

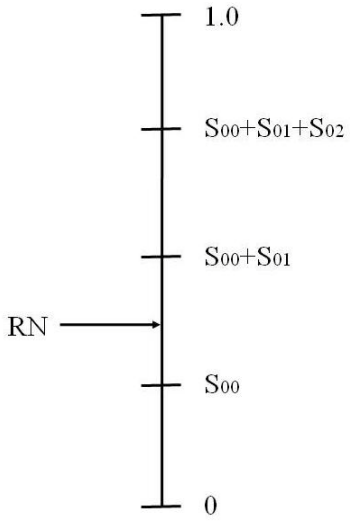


Figure 5.2 Random Number Generator

5.2.2. Scheme 2: Selecting a Path by Maximum Residual Bandwidth

As and alternative, in the source node we can try to select the path with maximum residual bandwidth among the four paths.

The procedure of selection of a path is as follows:

Choose the path for which residual bandwidth = max { B_{00}, B_{01}, B_{02}, B_{03} }

then the flow is routed along the chosen path.

5.2.3. Updating QoS State Information of the Network

When the flow arrives at the source node, we assume the parameters as in Figure 5.3.

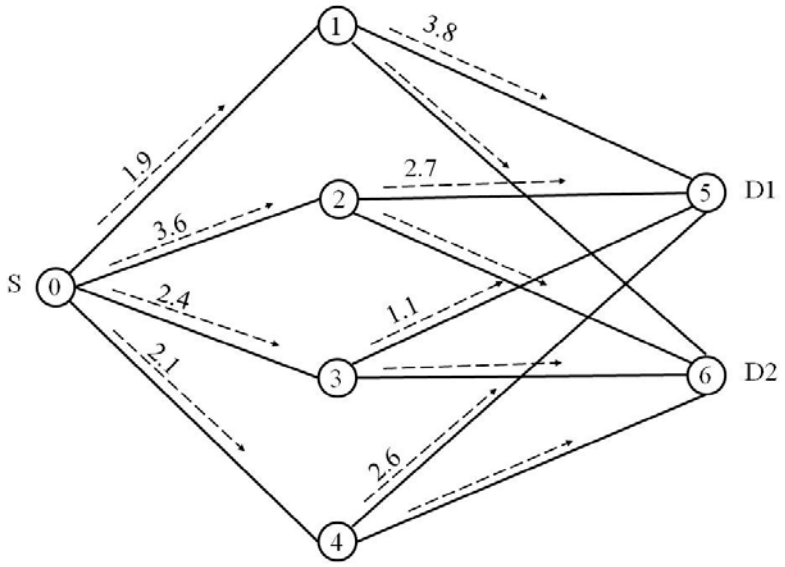


Figure 5.3 An example when a flow arrives to the source node

In Figure 5.3, the arrival flow required QoS bandwidth is 1.3, and its destination is D1 (Node 5). The path residual bandwidths are B_{00} =1.9, B_{01} =3.6, B_{02} =2.4, B_{03} =2.1, B_{10} =3.8, B_{20} =2.7, B_{30} =1.1 and B_{40} =2.6.

(1) In the case of scheme 1, the score of Path 00 $S_{00} = \frac{1.9}{1.9+3.6+2.4+2.1} = 0.19$, the score of Path 01 $S_{01} = \frac{3.6}{1.9+3.6+2.4+2.1} = 0.36$, the score of Path 02 $S_{02} = \frac{2.4}{1.9+3.6+2.4+2.1} = 0.24$ and the score of Path 03 $S_{03} = \frac{2.1}{1.9+3.6+2.4+2.1} = 0.21$.

Then we select a path by Random Number Generator as in Figure 5.4, where $S_{00} = 0.19$, $S_{00} + S_{01} = 0.55$ and $S_{00} + S_{01} + S_{02} = 0.79$. If we assume that RN =0.35, then Path 01 is selected and the flow is routed along Path 01 to Node 2.

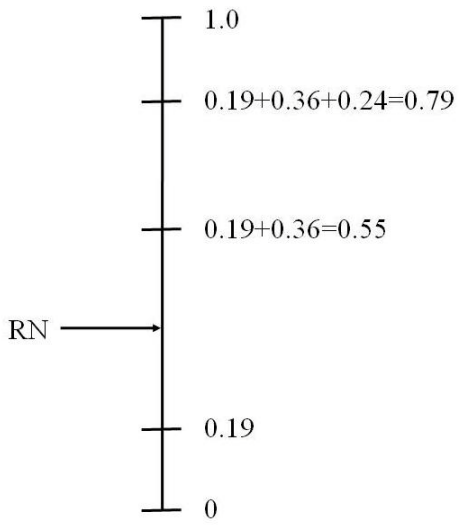


Figure 5.4 Selecting a path by Random Number Generator

At Node 2, when the flow arrives, there is only one candidate path to the destination D1. Then the flow is routed along Path 20 to Node 5.

(2) In the case of scheme 2, when the flow arrives at the source node,

max { B_{00}, B_{01}, B_{02}, B_{03}}= max { 1.9, 3.6, 2.4, 2.1}=3.6, then Path 01 is selected and the flow is routed to Node 2. The flow is then routed along Path 20 to Node 5.

(3) After the flow is routed from the source to the destination, the path residual bandwidths are changed as in Figure 5.5.

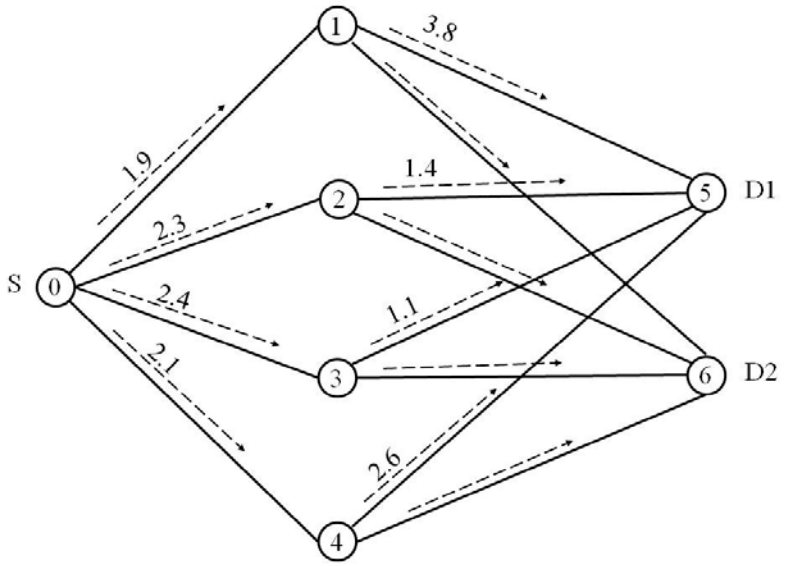


Figure 5.5 An example after the flow is routed to destination

In figure 5.5, B_{01} =3.6-1.3=2.3 and B_{20} =2.7-1.3=1.4 and the QoS state information of the network is updated.

When a new flow arrives at the source, the score of Path 00 is $S_{00} = \frac{1.9}{1.9+2.3+2.4+2.1} = 0.22$, the score of Path 01 $S_{01} = \frac{2.3}{1.9+2.3+2.4+2.1} = 0.26$, the score of Path 02 $S_{02} = \frac{2.4}{1.9+2.3+2.4+2.1} = 0.28$ and the score of Path 03 $S_{03} = \frac{2.1}{1.9+2.3+2.4+2.1} = 0.24$.

Scheme 1 will select a path by RN; scheme 2 will select a path using max {1.9, 2.3, 2.4, 2.1}.

5.2.4. Comparison between the two schemes

We now compare the two schemes using the example in Figure 5.5. When a new flow arrives at the source, we use the two schemes to select a path.

(1) Assume the new flow has required QoS bandwidth 2.35.

Scheme 1 selects Path 01 as the optimal path by Random Number Generator as shown in Figure 5.6, where RN =0.30. The flow is routed along Path 01. Since B_{01} =2.3 is less than the required QoS bandwidth 2.35, the flow is rejected.

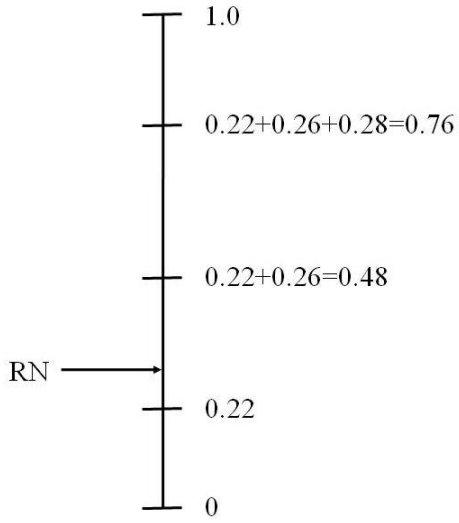


Figure 5.6 Selecting a new path by Random Number Generator

Scheme 2 selects Path 02 as the optimal path as max $\{B_{00}, B_{01}, B_{02}, B_{03}\}$= max {1.9, 2.3, 2.4, 2.1}=2.4. The flow is routed along Path 02 and then is accepted since B_{02}=2.4 is more than the required QoS bandwidth 2.35.

(2) Assume the new flow has required QoS bandwidth 1.2.

In scheme 1, Path 01 is selected as the optimal path by the Random Number Generator as displayed in Figure 5.6, where RN =0.30. The flow is routed along Path 01 to Node 2.

Since B_{01}=2.3 is more than the required QoS bandwidth 1.20, the flow is accepted. In Node 2, the flow is routed along Path 2. The flow is accepted as B_{20} =1.4 is more than the required QoS bandwidth 1.2.

In scheme 2, Path 02 is selected as the optimal path since max { B_{00}, B_{01}, B_{02}, B_{03}}= max {1.9, 2.3, 2.4, 2.1}=2.4. The flow is routed along Path 02 and then is accepted since B_{02} =2.4 is more than the required QoS bandwidth 1.2. When it arrives at Node 3, the flow is routed along Path 30. It is rejected since B_{30}=1.1 is less than the required QoS bandwidth 1.2.

The above indicates that between a source–destination pair, we have to take every link which the flow passes into account. The scheme 2 also only gives consideration to the current hop. Although the path with maximum residual bandwidth is selected to route the arrival flow, in the next hop, the path residual bandwidth may possibly be too small to fill the requirement of the flow required QoS bandwidth. The scheme 1 operates by a Random Number Generator. It is able to reflect the bandwidth state of the next hop. But, possibly, in the current hop, the path residual bandwidth is less than the flow required QoS bandwidth. Schemes 1 and 2, each have their strong points and shortcomings. Based on these ideas, two new algorithms are suggested in 5.3 and 5.4 below.

5.3. Score Based QoS Routing

According to scheme 1 above, we propose the localised Score Based QoS Routing algorithm (SBR). SBR relies on the variable of Scores which depends on the residual bandwidths among candidate paths. In the source node, the optimal path is selected by Random Number Generator. Only the path which is selected updates its information of

residual bandwidth after the flow is accepted. The source node stores every path residual bandwidth, and computes scores of candidate paths when a flow arrives.

The SBR procedure is as follows:

It is assumed that every node has a predetermined set of candidate paths to each possible destination.

1. with a flow arrival, check the first node along each candidate path to the destination
2. set score $S_i = \frac{B_i}{\sum_{i=1}^{k} B_i}$, where $k =$ number of candidate paths.
3. run Random Number (RN) Generator
4. if $RN \leq S_0$ then

   choose Path 0
5. else if $RN \leq S_0 + S_1$ then

   choose Path 1

   ……
6. else if $RN \leq S_0 + S_1 + \cdots + S_{k-2}$ then

   choose Path k-2
7. else choose Path k-1
8. route the flow along the chosen link to the next node.
9. is the link residual bandwidth $B_P \geq B_r$?
10. if no,

    reject the flow and inform the source node and end
11. else,

    accept the flow

12. update path P residual bandwidth $B_P = B_P - B_r$

13. is the current node the destination?

14. if no,

    go to step 1

15. else,

    end

In this procedure, S_i is the score of path $P_i \in R$, B_P is the path residual bandwidth, and B_r is the flow required QoS bandwidth.

Based on the view that a path with maximum residual bandwidth has the highest probability to be selected, SBR uses a Random Number Generator to choose the path. Each node stores the residual bandwidth of its outgoing links and updates this information when a flow is accepted on a path. In routers, SBR only requires a small memory space, and avoids more complicated calculations such as in PSR and CBR.

5.4. Bandwidth Based QoS Routing

In this section, we propose a scheme called Localised Bandwidth Based QoS Routing (BBR) which is based on scheme 2. In the same way as SBR, BBR stores every link residual bandwidth, and updates this information when a flow is accepted on a path. Within the set of candidate paths, the link with maximum residual bandwidth is selected as the next hop.

The BBR procedure is as follows:

1. with a flow arrival, check the first node along each candidate path to the destination
2. set $B = \max\{ B_1, B_2, \cdots, B_k \}$, $k =$ number of candidate paths
3. for $i = 1, \cdots, k$, is $B \geq B_r$?
4. if no,

   go to end
5. if yes,

   Path i is chosen and the flow is routed along the chosen link to the next node.
6. is the link residual bandwidth $B_i \geq B_r$? (B_r is the flow required QoS bandwidth.)
7. if no,

   reject the flow, send message to the source and go to end
8. else,

   accept the flow
9. update path residual bandwidth
10. is current node the destination?
11. if no,

    go to 1
12. else,

    end

BBR uses a simple method to compare the residual bandwidth of each path, and only the path with maximum residual bandwidth is selected. Its features are low memory loads and very simple calculations.

5.5. Performance Evaluation

We have proposed two localised QoS routing algorithms which are both bandwidth based - the Localised Score Based QoS Routing (SBR) and the Localised Bandwidth Based QoS Routing (BBR). In this section we evaluate the performance of these two algorithms. They are compared with CBR and Dijkstra's algorithm. Since the CBR algorithm has been shown to outperform the PSR algorithm [3], PSR is not used for comparison. Dijkstra's algorithm [115], which is a shortest path algorithm, takes at least $O(N \log N + L)$ time, where N is the size of the network measured as the number of nodes and L is the number of links. Dijkstra's algorithm always selects the most seemingly feasible path based on its current global state and keeps it until the next new updates arrive to correct the situation. Dijkstra(χ) represents Dijkstra's algorithm with update interval (χ), where Dijkstra (0) gives an optimum bound on performance since this represents a global algorithm with instantaneous updates, but is obviously unrealisable in practice. In addition to providing an optimum bound on performance, Dijkstra's algorithm is used as a basis for comparison since it forms the core of many contemporary source routing algorithms that use the global link state. In our simulations, these algorithms use bandwidth as the only metric for routing.

5.5.1. Simulator Configuration

Our aim is to simulate the arrivals of a large number of flows in order to generate accurate statistical results and simulate realistically modern networks which have enormous capacities and can accommodate a large number of flows. To assess performance, a simulator has been developed using OMNeT++ [86][87]. The simulator

has two levels, one is the flow-level, which selects routes according to a predetermined scheme and does admission control and resource reservation. The other one is the packet-level, which is developed to increase the level of realism by simulating connection setup and tear-down.

5.5.2. Network Topologies

In order to represent the underlying topology given the dynamic nature of current networks, we consider four networks, each representing a different topology with different characteristics in the simulation experiments. Random topologies are generated by the Brite generator which is a parametrized topology generation tool, and can be used to flexibly control various parameters (such as connectivity and growth models) and study various properties of generated topologies (such as power laws, average path length, etc) [103]. The ISP topology has appeared in different QoS routing studies [103][99]. A torus topology is also used in our simulations since this represents a more uniform topology. Table 5.1 lists the most important characteristics of the topologies used in the experiments.

Topology	Nodes	Links	Node degree	Avg. path length
RANDOM12	12	36	3	2.015
RANDOM60	60	184	3.067	5.176
ISP	50	138	2.760	4.281
Torus	80	320	4	4.557

Table 5.1: Topologies generated and their characteristics

In our simulation experiments, we assume that the network topology remains fixed for a specific experiment; links are bidirectional and have the same capacity C in each direction ($C = 150Mbps$). Every node can be both a source and a destination.

5.5.3. Simulation Setting

To simulate actual flow arrival intervals, we take both exponential and burst distributions into consideration. The traffic is generated at a source node according to a Poisson process with mean flow inter-arrival time $1/\lambda$, or a Weibull process with two different values of the shape parameter 0.3 and 0.7. The flow QoS requested bandwidth is uniformly distributed in the range between 0.1 to 2MB. The destination node is selected randomly with a uniform distribution, excluding the source node. The offered network load is $\rho = \lambda N\overline{bh}/\mu LC$, where N is the number of nodes, $\bar{b}$ is the average bandwidth required by a flow, $\bar{h}$ is the average path length (in number of hops) and L is the number of links in the network[116][117]. The parameters for the CBR algorithm are $MAX\_CREDITS = 5$ and $\Phi = 1$. Blocking probabilities are calculated based on the most recent 20 flows.

In the start of every simulation experiment, the sets of candidate paths are computed based on the current network topology. It is done for each sub-network and backbone separately with the topology information stored in an array of links and nodes. For the purpose of the simulation, the set of candidate paths between each source-destination pair in a selected network topology are chosen. The simulations of SBR and BBR choose the candidate paths by choosing the set to include all feasible paths. The simulations for CBR choose each candidate path set as the set of all minimum hop paths plus a set of alternative paths as (minimum hop)+1 paths to meet the needs. This

process starts by assigning an initial value 1 to all links in the network and then uses Dijkstra's algorithm to find the shortest candidate path. After finding the first candidate path, the weights on all the links along the path are increased, and the step to find the next candidate path is repeated until no more new paths can be found.

5.5.4. Performance Measures

Each experiment simulates the arrival of 2,000,000 flows with periodic increase of traffic loads. The simulation results are collected after every 200,000 flows. The first 200,000 flows are used to provide a run up period to allow the simulation to reach a steady state before any measurements are undertaken. With such a large number of flows in total (2,000,000), the 95% confidence intervals were found to be extremely tight and in most cases were not visible on the graphs with the scales used. They were subsequently dispensed with, both in this chapter and the subsequent ones. Since the performance of routing algorithms may vary across different load conditions, our simulation experiments consider a wide range of loads to assist in evaluating the algorithms under different load conditions. But in the case of low loads, flows are almost always accepted which results in very low blocking probabilities and all algorithms behave the same. Then we try to choose the most relevant range of loads and omit the rest since the relative performance of the algorithms is reflected in the chosen range, and the difference in the performance is otherwise insignificant.

We use two performance metrics to measure the performance of the algorithms: flow blocking probability and bandwidth blocking probability. To recap, they are defined as:

$$\text{flow blocking probability} = \frac{|B|}{|C|},$$

$$\text{bandwidth blocking probability} = \frac{\sum_{f \in B} bandwidth(f)}{\sum_{f \in C} bandwidth(f)},$$

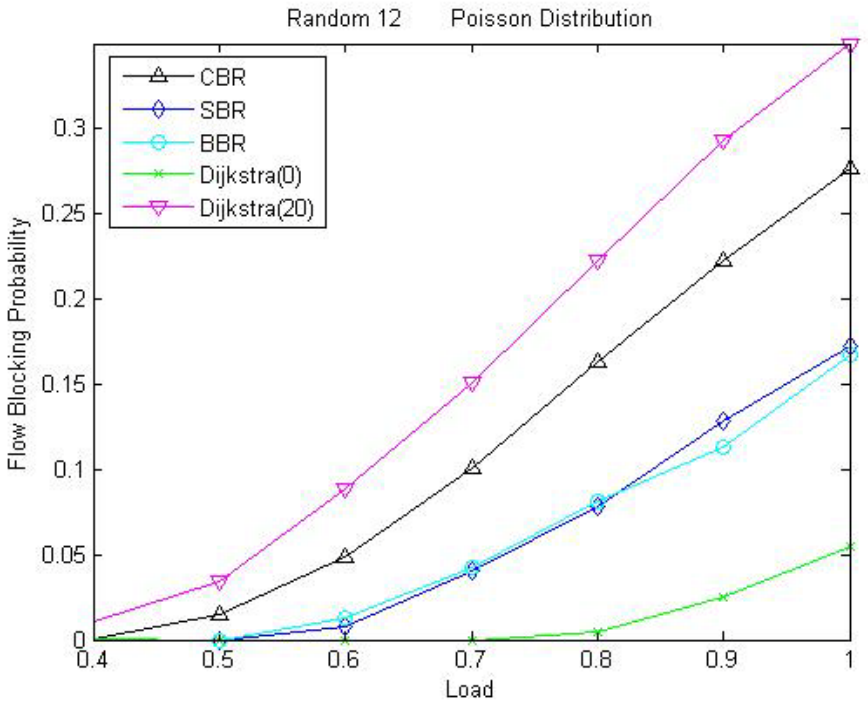
where B is the set of blocked flows and C is the set of the total flows and $bandwidth(f)$ is the requested bandwidth for path f.

5.5.5. Simulation Results

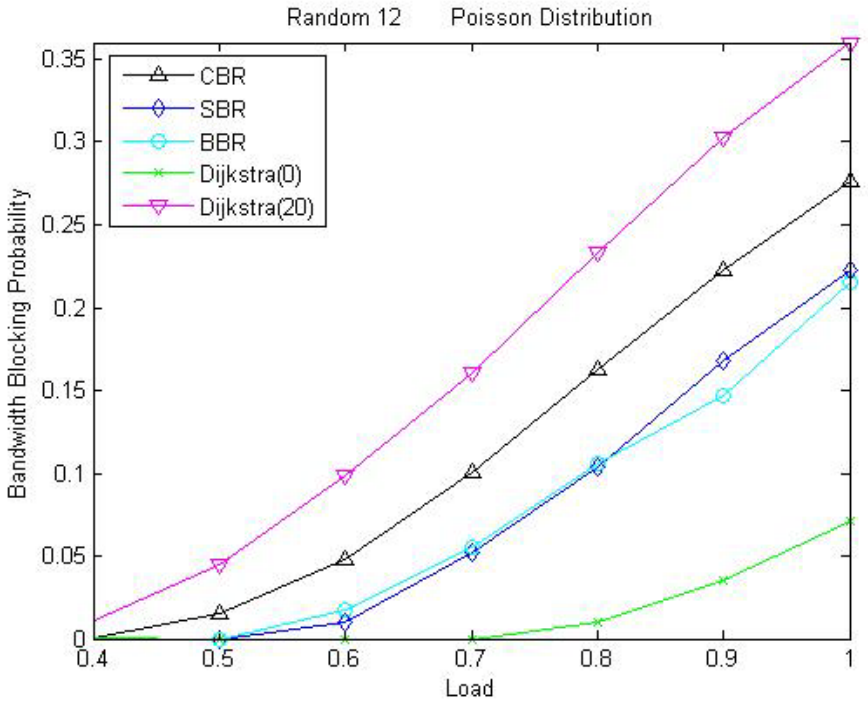
After a series of simulation experiments, the two blocking probabilities have been obtained and benchmarked against other existing algorithms.

5.5.5.1 Performance Comparison

The performance of the proposed SBR and BBR algorithms is compared with CBR and Dijkstra's algorithms as in Figure 5.7. The load is uniformly distributed among all the 12 nodes in a Random 12 topology and the overall flow blocking probabilities and bandwidth blocking probabilities under the different loads from 0.4 to 1.0 are plotted.



a Flow blocking probability



b Bandwidth blocking probability

Figure 5.7 Flow and bandwidth blocking probabilities for Random 12

The results in Figure 5.7 suggest the following:

(1) When loads are low (less than 0.4), the flow and bandwidth blocking probabilities of all the four algorithms are small, and their performance is similar. This is because the flows on every link are few, and the sum of their bandwidths is not more than the link capacity. Flows are therefore almost always accepted. However when loads increase, the flow and bandwidth blocking probabilities grow significantly and the difference among the performances becomes very marked.

(2) The update intervals clearly affect the performance of Dijkstra's algorithm significantly. Dijkstra(0) has the best performance as expected. Dijkstra(20) offers the worst performance; its flow and bandwidth blocking probability increases rapidly.

(3) The SBR and BBR performances are both better than CBR. Under small loads, the difference between them and CBR is insignificant but when loads increase, the difference becomes more apparent.

(4) The performance of SBR and BBR is almost the same for small and moderate loads, but BBR becomes better than SBR when loads increase.

To analyse the reasons for these performance differences, Dijkstra's algorithm is a global link state algorithm. It selects paths based on a QoS global state which is updated periodically. That is, Dijkstra's algorithm always selects the best seemingly feasible path based on its current global state and keeps it until the next new updates arrive to correct the situation. Dijkstra(X) means the Dijkstra's algorithm with update interval (X), where Dijkstra(0) gives an optimum bound on performance. When periodic updates increase and do not respond quickly enough to variations in network resources, the performance becomes worse and worse.

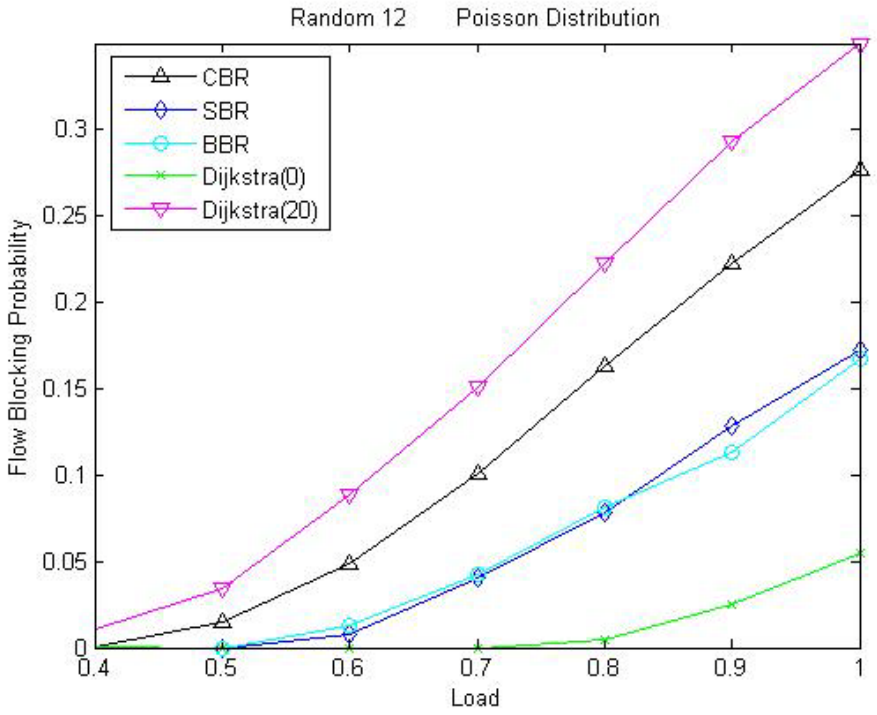
CBR selects the path with the maximum credits. The selected path credits are updated after every flow is routed along the path; any flow rejection will cause the decrease of

its credits and the selection of alternative path with more credits. Flow rejection is incurred by two factors: the flow required QoS bandwidth B_r and the path P residual bandwidth B_P. On a path, when the sum of the required QoS bandwidths B_r increases continuously to a value which is more than the path P residual bandwidth B_P, the flow is rejected. CBR only updates path credits after every flow is routed along that path. In other words, if no flow is routed along a path for a long time, the path credits will not be changed. At the time, it is possible that path P residual bandwidth B_P is changed as some other flows are routed over some links of the path and a number of bandwidths are released or reserved. Then a new flow routed along a path with high credits may possibly be rejected. In other words, indirect measures like path credits cannot reflect the actual current network QoS state fully.

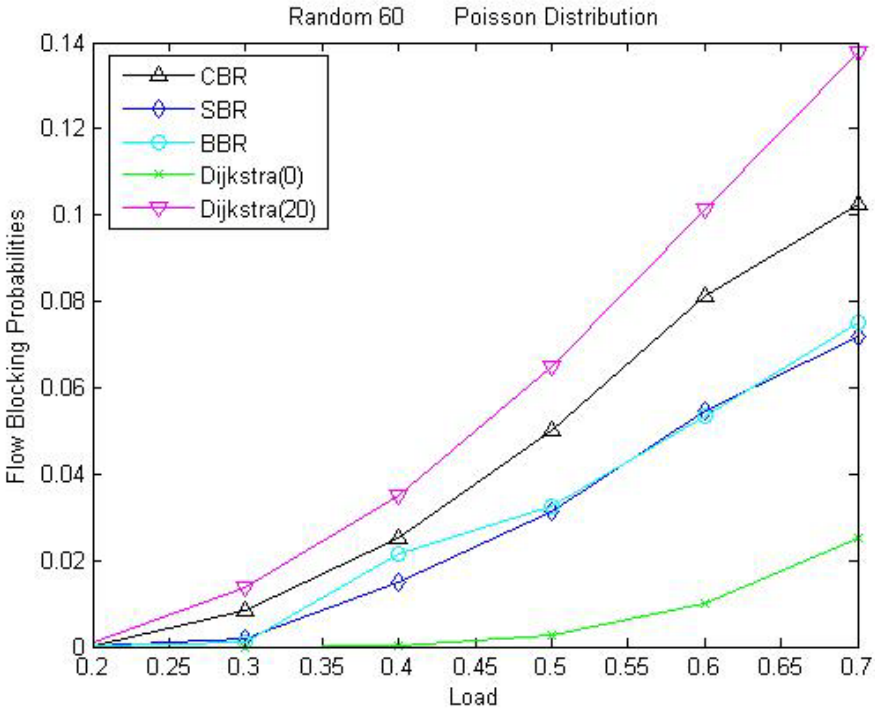
On the contrary, SBR and BBR see current path residual bandwidth B_P as an important factor. SBR selects a path by a Random Number Generator based on the view that a path with maximum residual bandwidth has the most probability to be chosen. BBR selects a path whose residual bandwidth is the maximum of the available paths. SBR and BBR are able to assess the actual network QoS state and so select a good path for every flow, if one exists. Under small and moderate loads, SBR performance is similar to BBR's. When loads increase, to get a path with maximum residual bandwidth is more important in the high traffic environment. Since the path SBR selects is not always the path with the maximum residual bandwidth, BBR can offer a better performance since BBR always attempts to select the best path.

5.5.5.2 Impact of Network Topologies

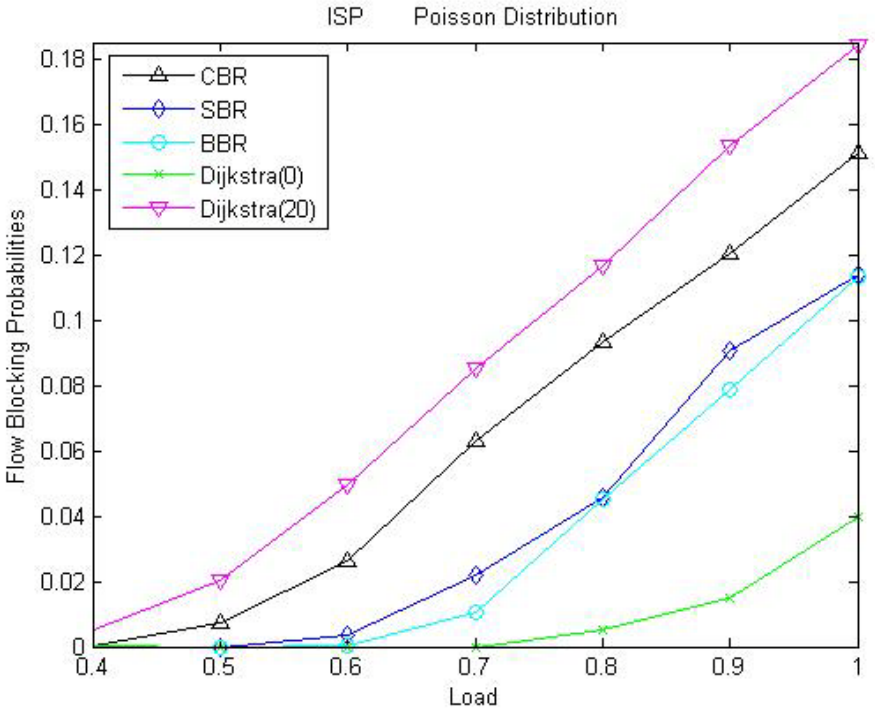
The performance of every routing algorithm may vary dramatically with the underlying network topology. We evaluate the performance of our algorithms using different types of network topologies. The flow blocking probabilities for the four schemes using the topologies previously are shown introduced in Figure 5.8.



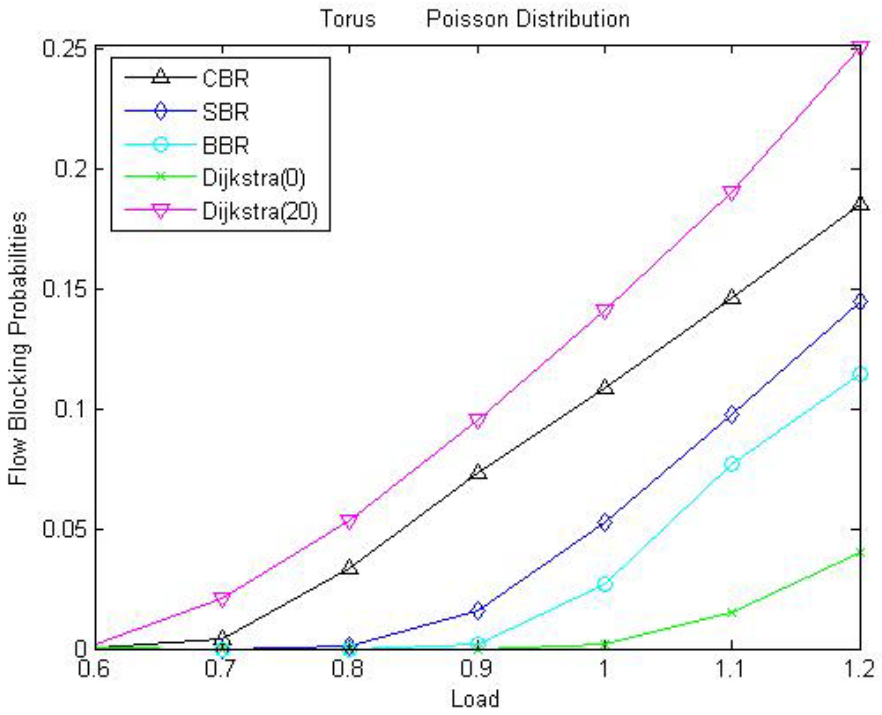
a Random topology RAND12



b Random topology RAND60



c ISP topology



d Torus topology

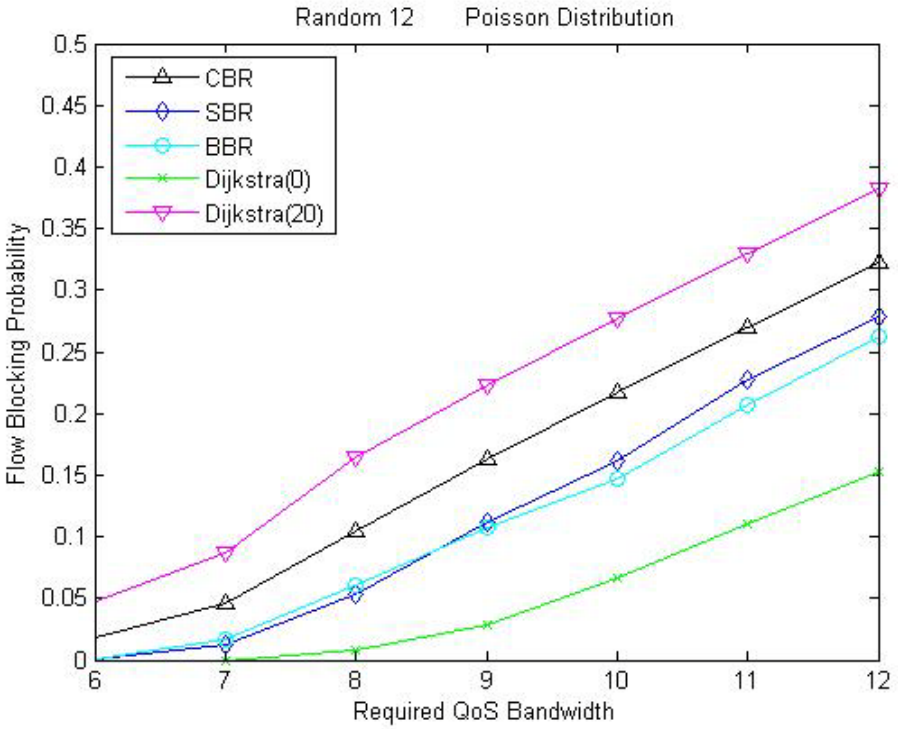
Figure 5.8 Impact of network topologies

From the figure, we find that in all cases the SBR and BBR schemes both perform better than the CBR scheme, Dijkstra(0) has the best performance, and Dijkstra(20) offers the worst performance. The difference between the performance of SBR and BBR and the performance of CBR is obvious. In the cases of RAND12 and RAND60 topologies, when load increases, the difference becomes bigger and bigger. However in the cases of ISP and Torus topologies, the difference gradually becomes closer and closer when load increases. With Random topologies the performance of SBR and BBR is similar, and the flow blocking probabilities both increase together when load increases. With the ISP topology the performance of SBR and BBR has some distinction, with BBR marginally outperforming SBR. With the Torus topology, BBR outperforms SBR much more significantly.

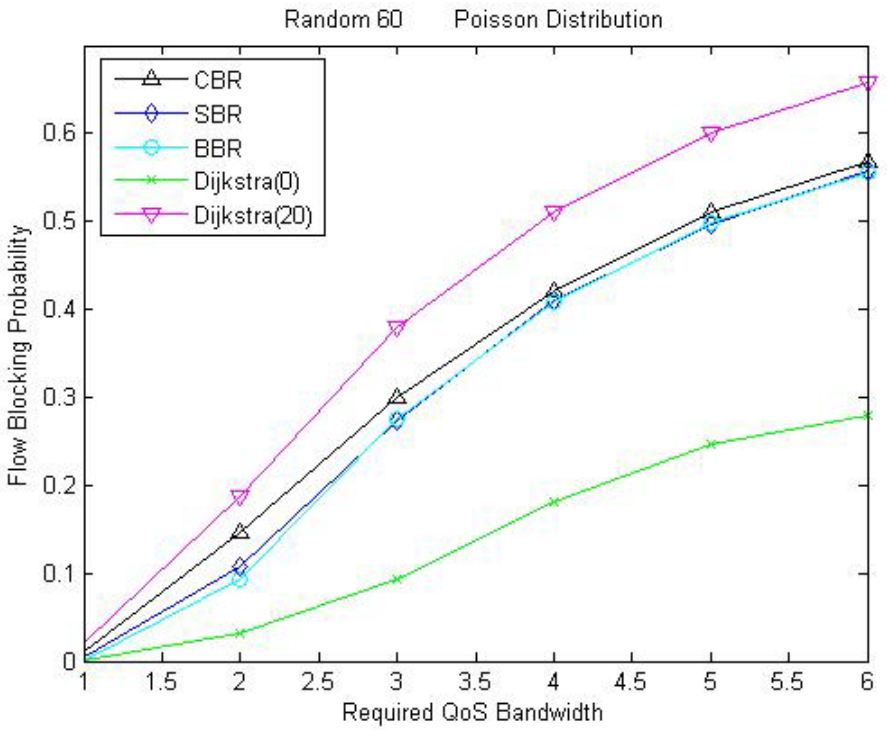
Through the comparisons we can conclude that SBR and BBR both have good performance for different types of network topologies, with the performance of BBR better than that of SBR for more uniform topologies.

5.5.5.3 Impact of QoS constraint

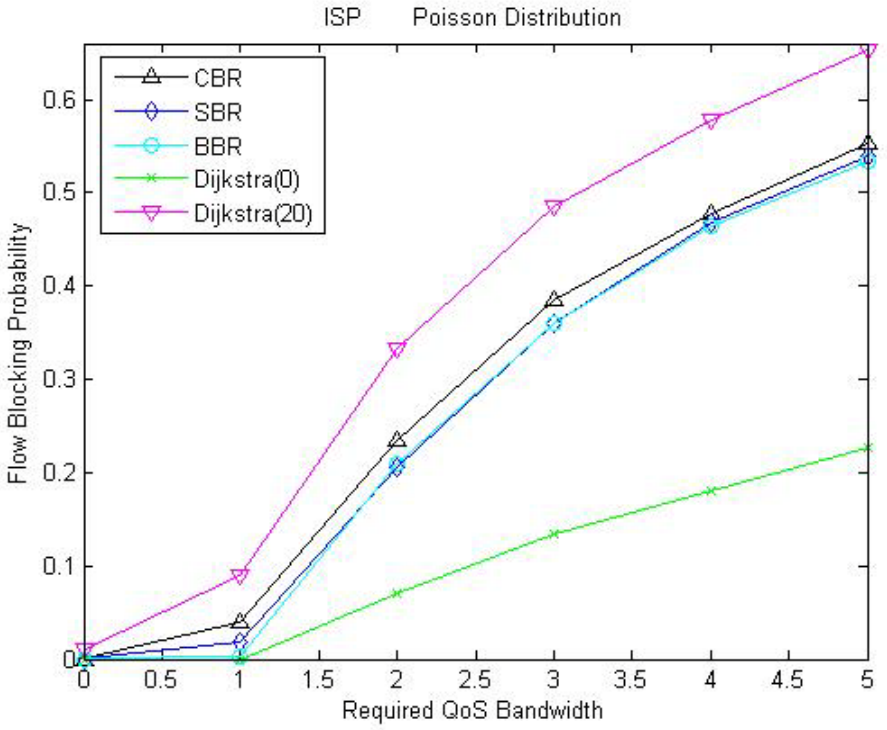
We have compared performance with different underlying network topologies against increasing load. Now we also compare them against increasing QoS constraint. In our simulation, we keep the load fixed as 0.7, where load 1 is set as the flow arrival intervals exponential(0.001) seconds; then we get results as in Figure 5.9, which plots the blocking probability for the different algorithms against the required QoS bandwidth, where the required QoS bandwidth l is set as 0.1 Mb/s.



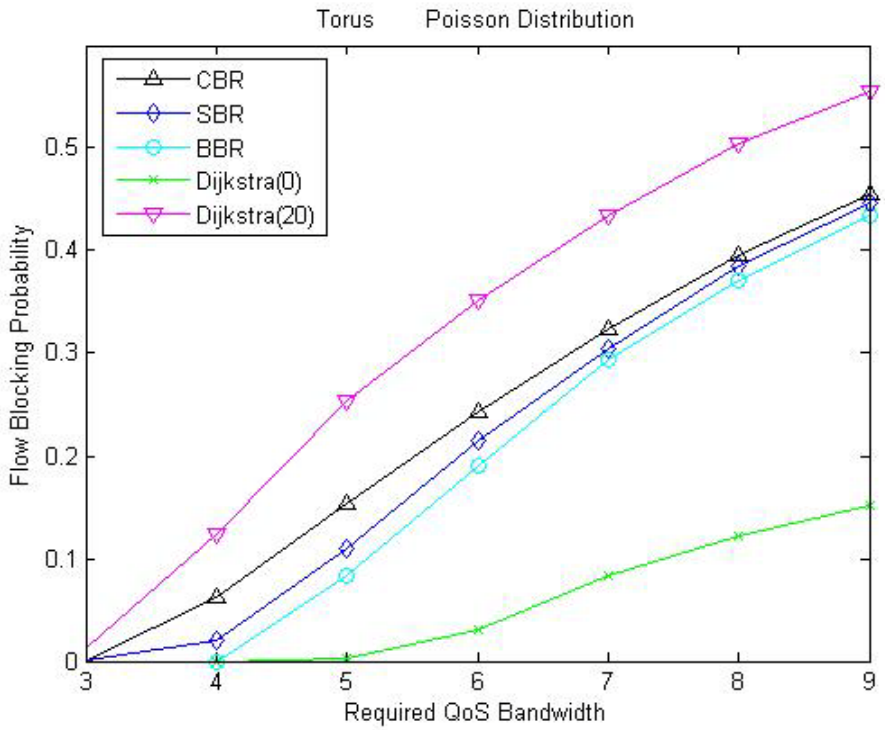
a Random topology RAND12



b Random topology RAND60



c ISP topology



d Torus topology

Figure 5.9 Impact of QoS constraint

Figure 5.9 demonstrates that as the required QoS bandwidth is increased then it becomes more difficult to satisfy the requirements and so the blocking probability increases. Against different required QoS bandwidths both of the SBR and BBR schemes perform better than the CBR scheme, Dijkstra(0) has the best performance, and Dijkstra(20) offers the worst performance. In the cases of RAND12 topology, when required QoS bandwidths are not more (less than 9), the blocking probabilities of BBR are higher than for SBR; but when required QoS bandwidths increase, the blocking probabilities of BBR become less than those of SBR. This means that the more the required QoS bandwidths increase, the better the performance of BBR is. In the cases of RAND60 and ISP topologies, the performance of SBR and that of BBR are very similar. They are all better than the performance of CBR. However in the case of Torus topology, the performance of BBR is still better than that of SBR for any of the required QoS bandwidths examined.

Through the comparisons we can conclude that SBR and BBR have good performance in the different types of network topologies examined against increasing required QoS bandwidth. On the whole, the performance of BBR is better than that of SBR.

5.5.5.4 Impact of Bursty Traffic

Some realistic traffic, such as IP traffic over Ethernets, has non-Poisson probability distributions and may actually show self-similar behaviour. It is important therefore that performance is also assessed when flow arrival is bursty and distributions have heavy tails [118][119]. So although Poisson traffic is widely used to model network flow arrivals, we also model bursty traffic in the Random 12 topology to evaluate the performance of our algorithms. To model the burstiness of traffic, we use a Weibull distribution with two different values of the shape parameter of the distribution, 0.3 and 0.7, where burstiness is increased with a smaller shape value [116][119]. Figure 5.10 shows the flow blocking probabilities plotted against the offered load from 0.2 to 0.8 with different shape values.

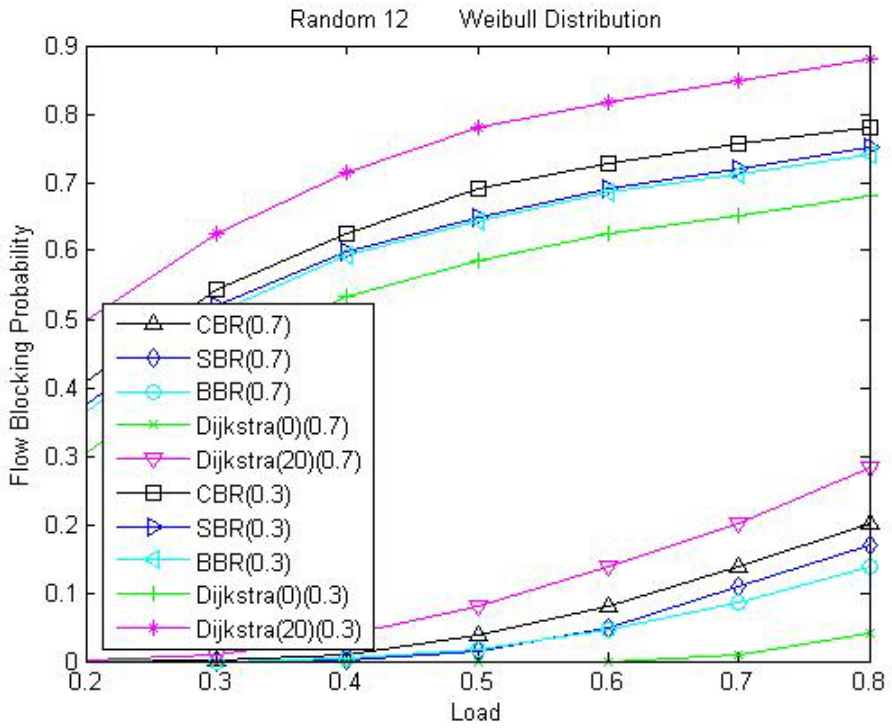


Figure 5.10 Impact of bursty traffic

From the figure, we note that the more burstiness in the network arrival, the higher the blocking probability of all the algorithms. The performance of Dijkstra(0) is the best performance, and Dijkstra(20) has the worst performance. The performance of SBR and BBR is better than CBR in the case of both shape 0.3 and 0.7. So SBR and BBR also have good performance in bursty traffic, with again BBR outperforming SBR. Poor performance resulting from high burstiness is perhaps expected since the periods of high traffic loads (bursts of traffic) will tend to swamp resources and so increase blocking.

5.5.5.5 Fairness

Having a fair share of resources is important where the resource demands of multiple flows sharing the resource are not met. In the absence of any other resource controls in the network, this means that there is at least one point along the end-to-end path where congestion is occurring, and we may determine how the resource is being shared by evaluating the resource distribution across the flows on that (part of) the path.

There are several well known definitions of fairness. Jain's Fairness Index (JFI) [104][105] is widely used for assessing system-wide fairness, which is defined as follows, $J = \frac{(\sum_{n=1}^{N} r_n)^2}{N\sum_{n=1}^{N} r_n^2}$, where, $\frac{1}{N} \le J \le 1$, N is the number of flows, r_n is the value of the resource attribute being assessed for flow n, e.g. r_n is the measured end-to-end throughput. $J = 1$ means there is fairness across all flows; $J = \frac{1}{N}$ indicates no fairness in the sense that just one flow consumes all the resources. When N is very large then J will tend to zero, i.e. $N \to \infty$, $J \to 0$. [106]

The Jain's Fairness Index for the three schemes is shown in Figure 5.11.

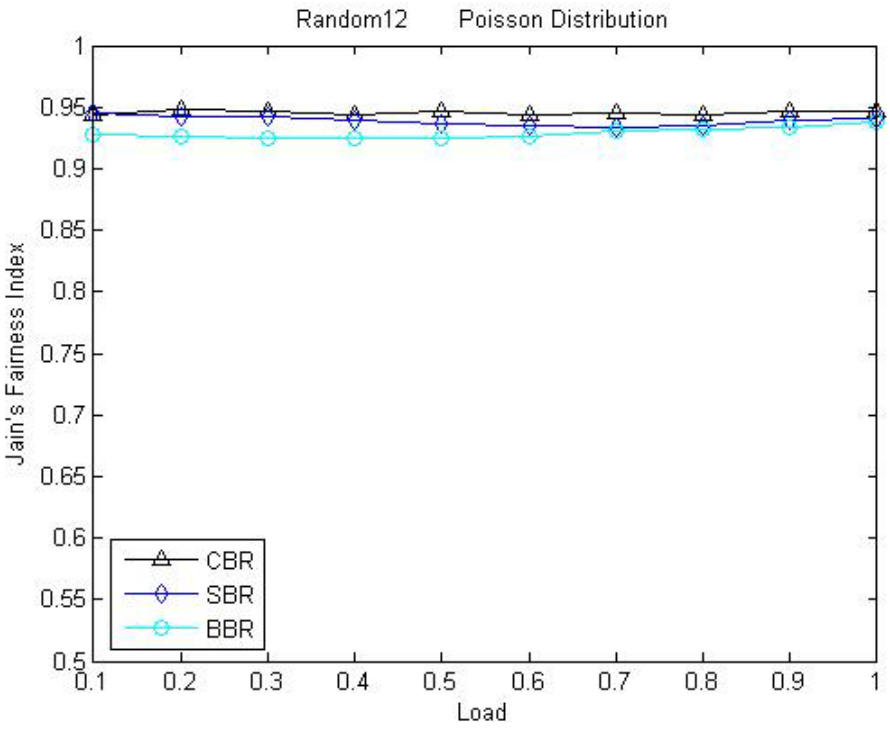


Figure 5.11 Jain's Fairness Index

Figure 5.11 demonstrates that CBR, SBR and BBR all offer good fairness. Their JFIs also appear invariant to load.

5.5.5.6 Nodal Storage, Communication Overhead and Time Complexity

In our simulations we have compared SBR and BBR with a current localised QoS routing algorithm (CBR) and a global QoS routing algorithm (Dijkstra's algorithm). Now we discuss the nodal storage requirements, overhead and time complexity of our suggested new algorithms.

First we note that Dijkstra's algorithm is used as a basis for a number of global QoS routing schemes to find the shortest or widest path. These take at least $O(N \log N + L)$ time, where N is the size of the network measured in the number of nodes and L is the

number of links. In addition, these global algorithms have huge communication overheads and storage requirements which have been previously well documented in the literature and discussed briefly in the thesis.

In contrast, if we now consider CBR, this selects a path among the set of candidate paths at the source node. Taking this operation in isolation, the worst case time complexity is simply $O(|R|)$, where R is the number of candidate paths. This has previously been quoted as the time complexity of both CBR and PSR[2][3]. Also, CBR requires to update the information of blocking probabilities, which takes a constant time $O(1)$. However, what appears to have been ignored in previous publications is that having selected a path, CBR must also check out its quality by sending a tentative path setup message along the path on a hop by hop basis. Thus, if the mean path length in hops, taken over all candidate paths is H, then the mean number of operations in setting up a path is Hk where k is a constant which arises from checking the residual bandwidth on the outgoing link of the chosen candidate path and determining if this meets the bandwidth requirement of the flow. Then as the network size grows without bound, the mean path length will also increase without bound and, assuming R remains finite, time complexity must become $O(H)$ and we see that it is therefore linear in path length. With regard to nodal storage for CBR, each node must store the sets of candidate paths to each possible destination. Assuming that all other nodes are possible destinations, this requires a storage of $RH(N-1)$ addresses at each node. This storage would therefore also grow linearly with network size.

Now to consider SBR and BBR, although the complete sets of candidate path must be computed at the outset as for CBR, because SBR and BBR both proceed on a hop by hop basis in a distributed fashion, only the addresses of the nodes at the end of the first hops need be stored in each node. Thus, with the same assumptions as for CBR, only

the $R(N-1)$ addresses of the one hop neighbours of the candidate paths need be stored at each node and not the full paths as in CBR. Then since the selection process among these R one hop neighbours must be carried out at each node in a candidate path, the number of operations required in setting up a path is HRk, where k has the same meaning as before. Thus time complexity is again $O(H)$, as for CBR.

We can conclude from this that localised QoS routing schemes are more generally scalable and incur far less communication overhead than global QoS routing schemes, with SBR and BBR the better choices among the localised QoS routing schemes.

5.6. Summary

In this chapter, after detailed discussion of bandwidth for QoS Routing, we proposed two schemes: the Localised Score Based QoS Routing (SBR) and the Localised Bandwidth Based QoS Routing (BBR) which all make routing decisions using residual bandwidth statistics collected locally. SBR selects a path by a Random Number Generator based on the view that a path with maximum residual bandwidth has the most probability to be chosen. BBR selects a path whose residual bandwidth is the maximum among the candidate path set. Their performance has been evaluated through a series of simulation experiments. We have compared SBR and BBR with CBR and the global QoS routing scheme, Dijkstra's algorithm, under different traffic loads and network topologies. We have demonstrated that the two proposed schemes perform better than CBR and Dijkstra's algorithm when this latter has a realistic update interval. In general our results suggest that:

- Localised QoS routing schemes which explicitly reflect the state of bandwidth of a path are better than schemes that are based indirectly on the path quality. CBR and

PSR are based indirectly on path quality through the credits and flow proportions respectively, but are not directly related to residual bandwidth, which is the required QoS metric.

- The scheme which selects a feasible path by comparing residual bandwidths directly is more effective than the scheme which uses a Random Number Generator to select a path based on relating residual bandwidth to selection probability. That is, BBR outperforms SBR in a majority of simulation experiments.
- It is demonstrated that the proposed algorithms both have good load balancing properties.
- The time complexity and communication overheads of localized schemes are generally far superior to global schemes.

This latter point implies that the localised schemes are generally much more scalable than equivalent global schemes.

Chapter 6

Localised Delay Based QoS Routing

6.1. Introduction

In the previous chapters of this thesis we proposed two efficient localised QoS routing algorithms which both use bandwidth as the only metric for routing. In this chapter we discuss delay for QoS routing and modify CBR to use delay instead of bandwidth as the metric. Then we propose Localised Delay Based QoS Routing (DBR) which relies on delay on the path in order to take routing decisions. DBR also is compared with other algorithms such as CBR and a global QoS routing scheme according to flow rejection probability under different network loads and topologies.

6.2. Consideration of Mean Delay for QoS Routing

The requirement here is that a flow between a source–destination pair is accepted or rejected according to whether its mean end to end delay is less than or equal to the required mean delay for that flow while at the same time not jeopardizing the QoS requirements of any existing flow. There is some argument as to whether mean end-to-end delay can truly be classed as a QoS metric since it cannot give a guarantee on instantaneous end-to-end delay such as might be required by some interactive real-time services. However, because of the asynchronous nature of a packet switched network,

instantaneous delay is a random variable (varies from instant to instant) and so cannot be supported as a QoS metric in such an environment. However, we argue that mean delay can be used as a constraint relating to the QoS and so be useful for services in which prompt delivery is desirable. Figure 6.1 shows a simple network with seven nodes and is used to illustrate the above requirements and their consequences. In what follows, we refer to mean delay simply as delay.

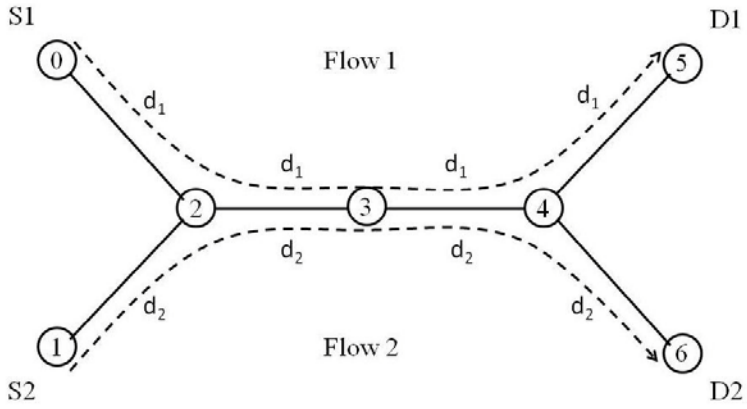


Figure 6.1 An example of mean delay for routing

In Figure 6.1, flow 1 is routed from source node 0, through nodes 2, 3 and 4, to destination node 5, where node 0 and node 5 are marked respectively as $S1$ and $D1$. Arriving to each node in the path, it is assumed that flow 1 adds delay d_1 to the node. Correspondingly, flow 2 is routed from $S2$, the source node 1, through nodes 2, 3 and 4, to $D2$, the destination node 6. It arrives to every node with delay d_2. Flow 1 and Flow 2 share links from node 2 to node 4. It is assumed that the required QoS delay $D_r = 9$ for both flows and $d_1 = 2$, $d_2 = 1$. In the case that there is only flow 1 on the network, since the flow cumulative delay $D(S1 - D1) = 4d_1 = 8$, then $D(S1 - D1) < D_r$, and flow 1 is accepted. Now if a new flow 2 arrives on the network,

$D(S1-D1) = d_1 + (d_1 + d_2) + (d_1 + d_2) + d_1 = 2 + (2+1) + (2+1) + 2 = 10$ and now, $D(S1-D1) > D_r$, and so flow 2 would be rejected, even though we can see that $D(S2-D2) = d_2 + (d_2 + d_1) + (d_2 + d_1) + d_2 = 1 + (1+2) + (1+2) + 1 = 8 < D_r$ since if we accept flow 2 this would jeopardize the required QoS of the existing flow 1.

Two conclusions are drawn from the above example: (1) Each flow is accepted if its cumulative delay is less than or equal to its required QoS delay; (2) Any new flow will affect the cumulative delays of existing flows which share common links, and so may jeopardize existing flows unless the new flows are rejected. Thus it is necessary to use some form of admission control in conjunction with any routing algorithm that uses mean delay as the QoS metric. Note that this situation does not arise when using residual bandwidth as the QoS metric since while mean end to end delay is an additive, path based metric, bandwidth is a concave, link based metric. That is, when using bandwidth as the metric, a QoS routing algorithm only has to check if each separate link of a chosen path has sufficient residual bandwidth to accommodate a new flow to determine whether or not the new flow can be accepted, irrespective of how many existing flows are sharing a link since existing flows are accounted for in the bandwidth used already used up on the link.

6.3. Description of the Delay Based Routing Algorithm (DBR)

The key rationale behind DBR is based on the observation that if each node maintains a set of candidate paths to all other nodes in the network, then if in choosing a path between a specific source and destination we proceed one hop at a time by choosing the next node as the one with the lowest mean delay among the relevant candidate path set,

then since the next node must also have a set of candidate paths to the required destination, we can simply repeat the same procedure from the next node and so on until the next node is the destination. The path selection and set up procedure is therefore distributed, unlike both PSR and CBR which select a specific end to end path at the outset based the computed flow proportions or path credits respectively. Also, note that the flow proportions or credits can have been calculated based on stale information. With DBR, the path is not selected at the outset but is chosen on a hop by hop basis based on the current mean delay at the nodes.

The details as to how DBR operates are as follows: Every node in the network is assumed to be able to act as a router that has a single shared buffer and is associated with a variable D_i which is called the node mean delay, where D_i is the mean delay of a packet through node i. Each packet must belong to a specific flow routed through that node and it is therefore clear that each packet must encounter the same mean delay through a specific node, irrespective of the flow to which it belongs. Each node is also required to maintain a predetermined set of R candidate paths to each possible destination as with SBR and BBR in the previous chapter. When a path selection and set up for a new flow arrives to a node, DBR chooses the hop to the next node by choosing the node with minimum mean delay among the set of candidate paths R. That is to say, after comparing all D_i which belong to the candidate paths' next nodes, only the path for which next node D_i is the minimum is chosen as the tentative route. If $D > D_r$, the flow is rejected, where D is a variable called the flow cumulative delay, which is the sum of the flow delays up to the present position, and D_r is the flow required QoS delay. Else when $D \le D_r$, the admission control is called. If accepting this flow causes

any existing flow's actual cumulative delay to exceed its required delay D_r, the flow is rejected, otherwise the flow is accepted.

Under the DBR scheme, routers only need to maintain and store sums of delays of flows on the nodes plus their candidate path sets. Also, the only tasks that the routers have to carry out upon receiving a request to route a flow to a specific destination is to send a multicast message to each of the R next nodes in the candidate path set to that specific destination requesting the nodes' mean delays, compare them and then select the node with the least mean delay, check that using this as a next node will not violate the QoS requirements of any existing flows through this node and then tentatively reserve this link in the path if the path acceptance criteria are satisfied or otherwise send a reject message to the source node. Any complicated calculations are thus avoided.

The following is a pseudo-code for DBR:

1. Set node mean delay D_i = sum of delay of flows on the node i
2. Among the R candidate paths to the destination, choose the next node as the node with the least mean delay.
3. Is the flow cumulative delay $D \le D_r$?
4. If no
5. reject the flow
6. Else
7. Does accepting the flow cause the end to end delay of any existing flow exceed its required delay D_r ?
8. If no
9. accept the flow
10. Else

11. reject the flow
12. Repeat the process from step 2 until the next node is the destination node

6.4. Performance Evaluation

In this section we evaluate the performance of our proposed algorithm - Localised Delay Based QoS Routing. We compare DBR with CBR and Dijkstra's algorithm. Since the CBR algorithm has been shown to outperform the PSR algorithm,[3] PSR is not used for comparison. Dijkstra's algorithm [115], which is a shortest path algorithm, takes at least $O(N \log N + L)$ time where N is the size of the network measured as the number of nodes. Dijkstra's algorithm always selects the most seemingly feasible path based on its current global state and keeps it until the next new updates arrive to correct the situation. Dijkstra(x) means the Dijkstra's algorithm with update interval (x), where Dijkstra(0) gives an optimum bound on performance as before. In our simulations, all of these algorithms use delay as the only metric for routing. We again use OMNeT++[86][87] as the simulation platform.

6.4.1. Network Topologies

For reflecting the underlying topology given the dynamic nature of current networks, we consider four networks which represent different topologies with different characteristics in our simulation experiments. These are identical to these described and used in Chapter 5. For convenience, Table 6.1 lists the most important characteristics of the topologies used in the experiments.

Topology	Nodes	Links	Node degree	Avg. path length
RANDOM60	60	184	3.067	5.176
RANDOM12	12	36	3	2.015
ISP	50	138	2.760	4.281
Torus	80	320	4	4.557

Table 6.1: Topologies generated and their characteristics

In our simulation experiments, we assume that the network topology remains fixed; links are bidirectional and have the same capacity in each direction. Every node can be both a source and a destination.

6.4.2. Simulation Setting

To simulate actual flow arrival intervals, we take both exponential and burst distribution into consideration. Flows are considered to arrive to each node with delays which are exponentially distributed with mean value 0.0005 sec. That is, each flow adds an exponentially distributed delay with mean 0.0005 sec. to each node through which it passes. Destinations are selected randomly with a uniform distribution. For the purpose of the simulation process, we assume that all of flows have the same required QoS delay $D_r = 1$ sec. Simulations of DBR, CBR and Dijkstra's algorithm all use mean delay as the only metric for routing. In other words, these schemes select paths and accept or reject flows according to the variables of the flow and node mean delays. The offered network load is $\rho = \lambda N \overline{bh} / \mu LC$, where N is the number of nodes, $\bar{b}$ is the average

bandwidth required by a flow, $\bar{h}$ is the average path length (in number of hops) and L is the number of links in the network[116][117]. Note that when the network load ρ is varied, since for a given simulation on a given topology N, $\bar{b}$, $\bar{h}$, L and C are all fixed quantities so the only variables are λ and μ. We must therefore vary λ/μ accordingly to give a specific load. Note also that for different topologies, the mean path length ($\bar{h}$) changes (see Table 6.1) so λ/μ will change with the topology to give the same load. The parameters for the CBR algorithm are $MAX\_CREDITS = 5$ and $\Phi = 1$. Blocking probabilities are calculated based on the most recent 20 flows.

At the start of every simulation experiment, the set of candidate paths is computed based on the current network topology. This is done for each sub-network and backbone separately with the topology information stored in an array of links and nodes. For the purpose of the simulation, the set of candidate paths between each source-destination pair in a selected network topology are chosen. The simulation of DBR requires the set to include all feasible paths. In the simulation of CBR, the set includes minhop paths and alternative paths. Between each pair, paths of minimum hop and (minimum hop)+1 are chosen to meet the needs. This process starts by assigning an initial value 1 to all links in the network and then uses the Dijkstra's algorithm to find the shortest candidate path. After finding the first candidate path, the weights on all the links along the path are increased, and the step to find the next candidate path is repeated until no more new paths can be found.

6.4.3. Performance Measures

The arrival of 2,000,000 flows is simulated, and the simulation results are collected after the first 200,000 flows. Since the performance of routing algorithms may vary

across different load conditions, our simulation experiments consider a wide range of loads to assist in evaluating the algorithms under different load conditions. But in the case of low loads, flows are almost always accepted which results in very low blocking probabilities and all algorithms behave the same. Then we try to choose the most relevant range of loads to reflect the relative performance of the three algorithms used.

The flow blocking probability is an important performance metric which is defined as:

$$\text{flow blocking probability} = \frac{|B|}{|C|},$$

where B is the set of blocked flows and C is the set of the total flows.

6.4.4. Simulation Results

We have obtained simulation results of DBR, CBR and Dijkstra's algorithm in various networks with different flow arrival intervals.

6.4.4.1 Performance Comparison

The performances of the three algorithms are compared in Figure 6.2. The overall flow blocking probabilities under the different load conditions are plotted for the random 60 topology. The x in Dijkstra(x) denotes the relative length of the link state update interval as before.

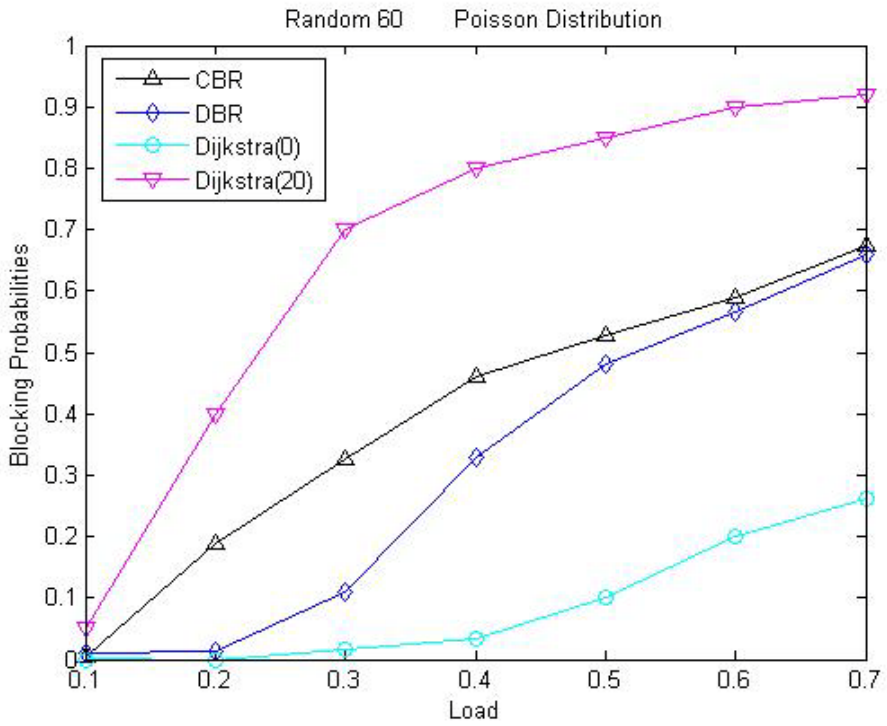


Figure 6.2 Flow blocking probabilities

Referring to Figure 6.2, we can observe the following:

(1) When loads are very low, the flow blocking probabilities of all three algorithms are small and their performances are similar. This is because flow cumulative delays are low and so flows are almost always accepted. When loads increase, the flow blocking probabilities grow significantly and the difference among the performances is very marked.

(2) The update intervals significantly affect the performance of Dijkstra's algorithm. Dijkstra(0) has the best performance. Dijkstra(20) offers the worst performance; its flow blocking probability increases rapidly, even under small and moderate loads.

(3) The DBR performance is generally better than that of CBR. Under small and moderate loads, the difference between them is very marked and when loads increase,

even though DBR flow blocking probabilities are close to CBR, DBR still exceeds the CBR performance.

(4) When loads are very high the performances of all algorithms would be expected to converge again close to a blocking probability of 1 due to no paths being available that satisfy the QoS.

To analyze the above, Dijkstra's algorithm is a global link state algorithm. It selects paths based on a QoS global state which is updated periodically. That is, Dijkstra's algorithm always selects the best seemingly feasible path based on its current global state and keeps it until the next new updates arrive to correct the situation. Since Dijkstra(x) means Dijkstra's algorithm with update interval (x), then Dijkstra(0) must give an optimum bound on performance since this implies instantaneous update using a global link state, which is, of course, nowhere near attainable in practice. When periodic updates increase and do not respond quickly enough to variations in network resource utilization, the performance becomes worse and worse.

CBR selects the path with the maximum credits. The selected path credits are updated after every flow is routed along the path; any flow rejection will cause the decrease of its credits and the selection of an alternative path with more credits. Flow rejection is incurred by two factors: the flow cumulative delay D and the node mean delay D_i. When the flow cumulative delay D increases continuously to a value which is more than its required QoS delay D_r, the flow is rejected. Each node adds its node mean delay D_i to every flow cumulative delay D when the flow is routed through the node. The key thing here is that CBR only updates path credits after every flow is routed along that path. In other words, if no flow is routed along a path for a long time, the path credits will not be changed. In the meantime, it is possible that the delays on the path

will change and so the credits can become stale in a similar way to an outdated link state due to a long update interval.

On the contrary, DBR sees the node instantaneous mean delay D_i when selecting a path and so always operates with up-to-date information.

We will further illustrate this drawback of CBR through the example in Figure 6.3.

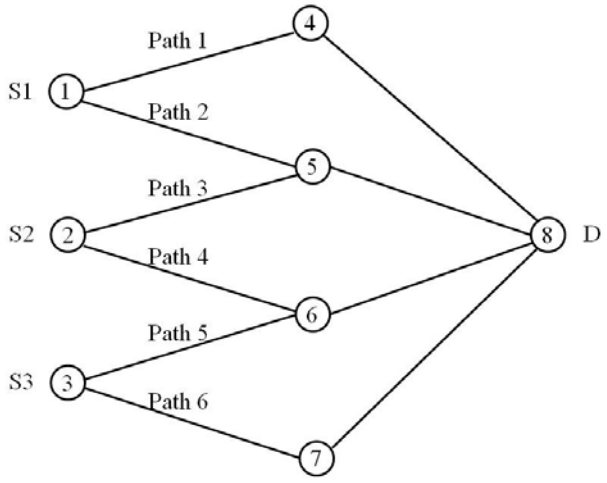


Figure 6.3 An example

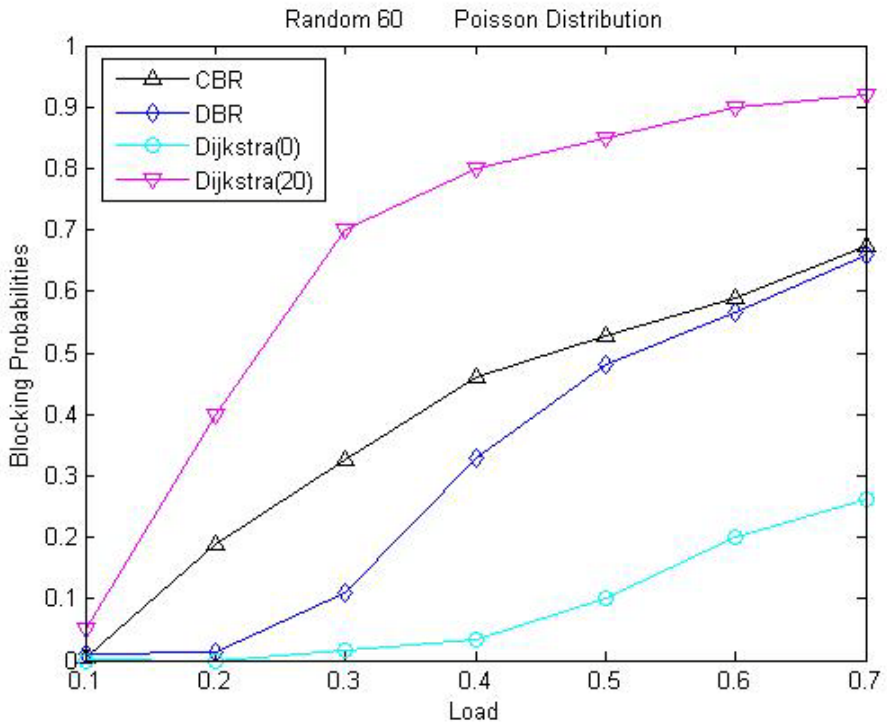
In the network in Figure 6.3, assume that flows are routed respectively from the three sources, $S1$ (Node 1), $S2$ (Node 2) and $S3$ (Node 3), to the destination D (Node 8) continuously and that for all flows $D_r = 1$. Every source has two alternative paths. When a flow arrives, the source node uses an algorithm to select a path and routes the flow along it. To first consider CBR, assume that after some time when flows have been routed from the three sources to their destination, Path 3 credits $P3.credits = 4.0$ and Path 4 credits $P4.credits = 3.0$, while Node 5 mean delay $D_5 = 0.5$, and Node 6 mean delay $D_6 = 0.6$. Assume that after a further period during which no flows arrive to $S2$ (Node 2), some flows arrive to $S1$ (Node 1) and $S3$ (Node 3), and are routed along

Path 2 and Path 5. The credits of Path 3 and Path 4 remain unaltered in that $P3.credits = 4.0$ and $P4.credits = 3.0$, but Node 5 and Node 6 mean delays were changed to $D_5 = 0.8$ and $D_6 = 0.7$ since flows were routed to them along Path 2 and Path 5. When a new flow arrives to $S2$ (Node 2), CBR must select Path 3 as $P3.credits > P4.credits$, and route the flow along it. Then when the flow with the cumulative delay $D = 0.25$ arrives at Node 5, the flow cumulative delay is added as $D + D_5 = 0.25 + 0.8 = 1.05$, which is more than the required QoS delay $D_r = 1$. The flow would therefore be rejected. If we now consider how DBR would behave, DBR selects Node 6 as the next hop since $D_6 < D_5$, and routes the flow along Path 4. At Node 6, the flow cumulative delay is added as $D + D_6 = 0.25 + 0.7 = 0.95$ which is less than the required QoS delay $D_r = 1$. The flow is therefore accepted. Thus CBR is required to select a new alternative path while DBR was able to choose this alternative at the first attempt to directly satisfy the required QoS.

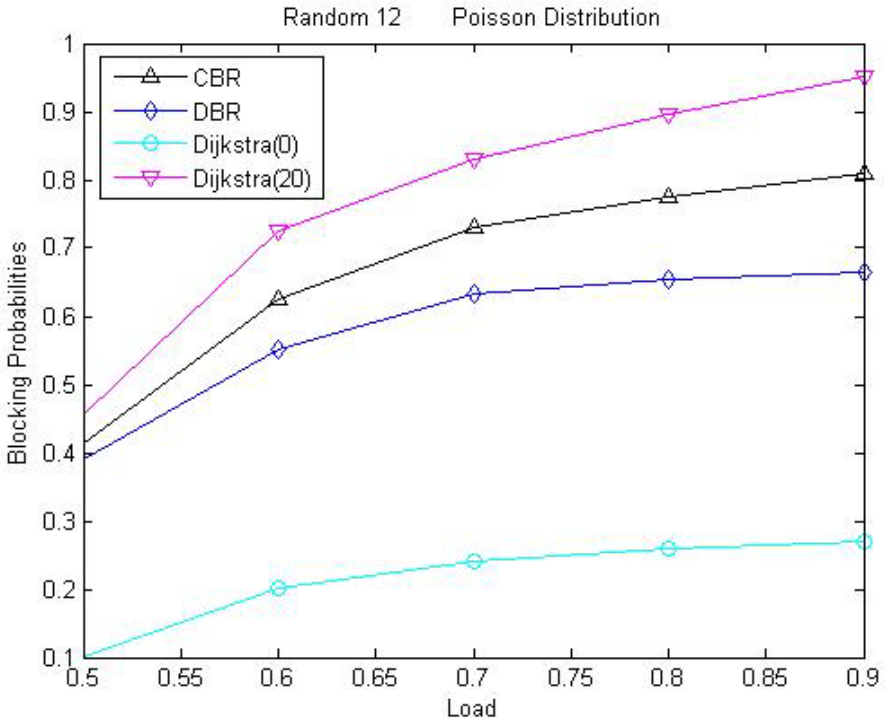
In terms of our discussion above, the results in Figure 6.2 can be summarized as follows. Dijkstra's algorithm with update interval 20 offers the worst performance due to its inability to react in a timely manner resulting from its long update period. Benefiting from its credit based QoS routing scheme, CBR performs better than Dijkstra(20). However CBR only updates credits when the paths are used and therefore cannot be relied on to reflect the actual network QoS state correctly. DBR circumvents the problems associated with the above two schemes and its performance is better than those of both CBR and Dijkstra(20). As a localised QoS routing algorithm, DBR has much lower communication overhead since it does not have to collect, update and distribute information relating to the global QoS network state. As a reference, Dijkstra(0) provides an optimum bound on performance.

6.4.4.2 Impact of Network Topologies

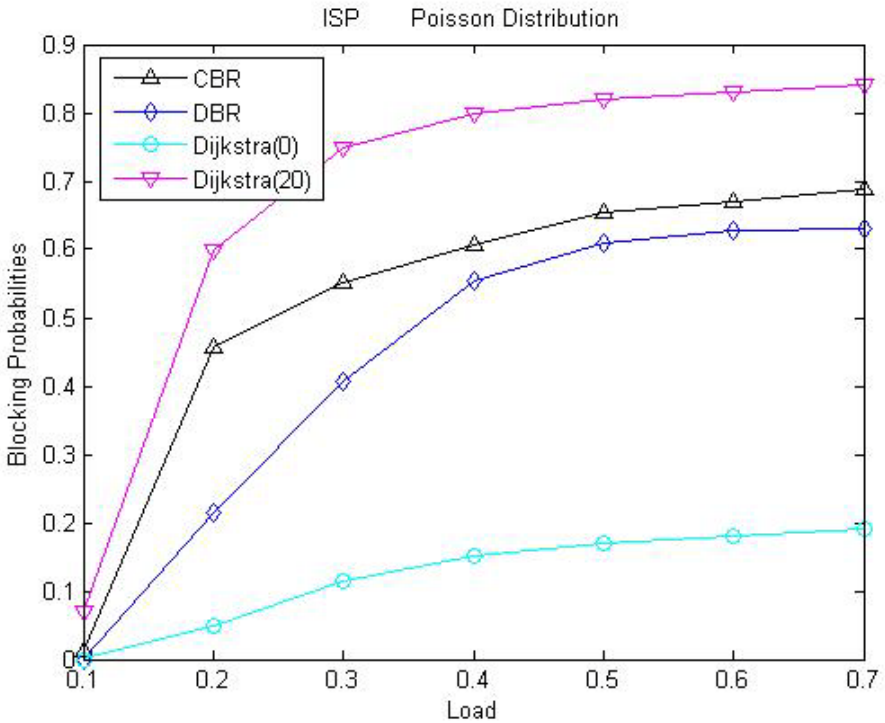
Since the performance routing algorithms may vary dramatically with the underlying network topology, in our simulation experiments the performances of the algorithms are evaluated using the random, ISP and torus topologies. The flow blocking probabilities for the four algorithms are shown in Figure 6.4, where subfigures a, b, c and d are the simulation results for the four topologies with a Poisson flow arrival distribution.



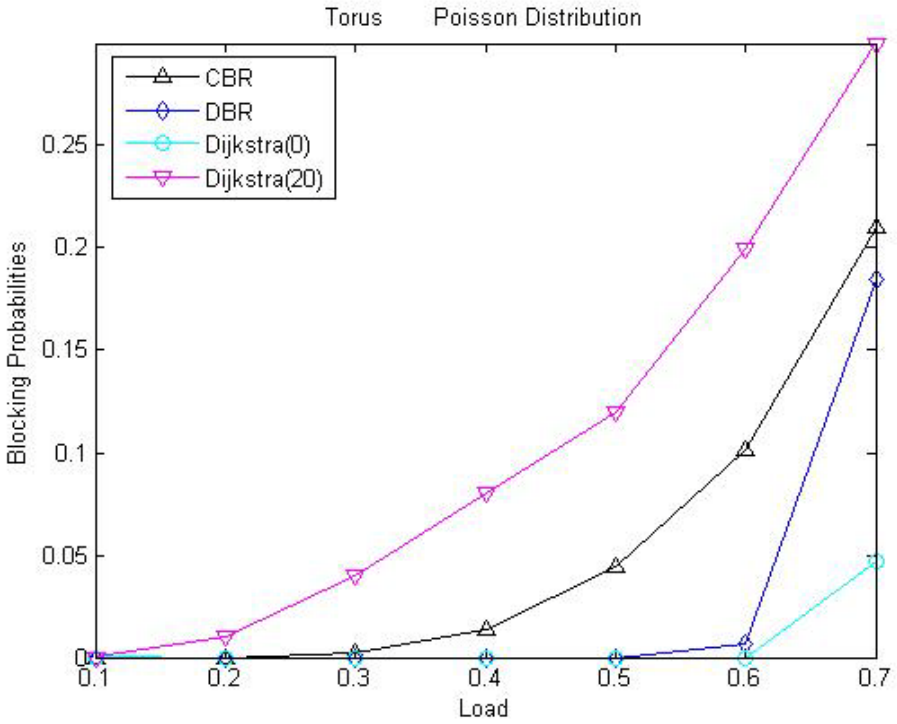
a Random 60 topology with Poisson distribution



b Random 12 topology with Poisson distribution



c ISP topology with Poisson distribution



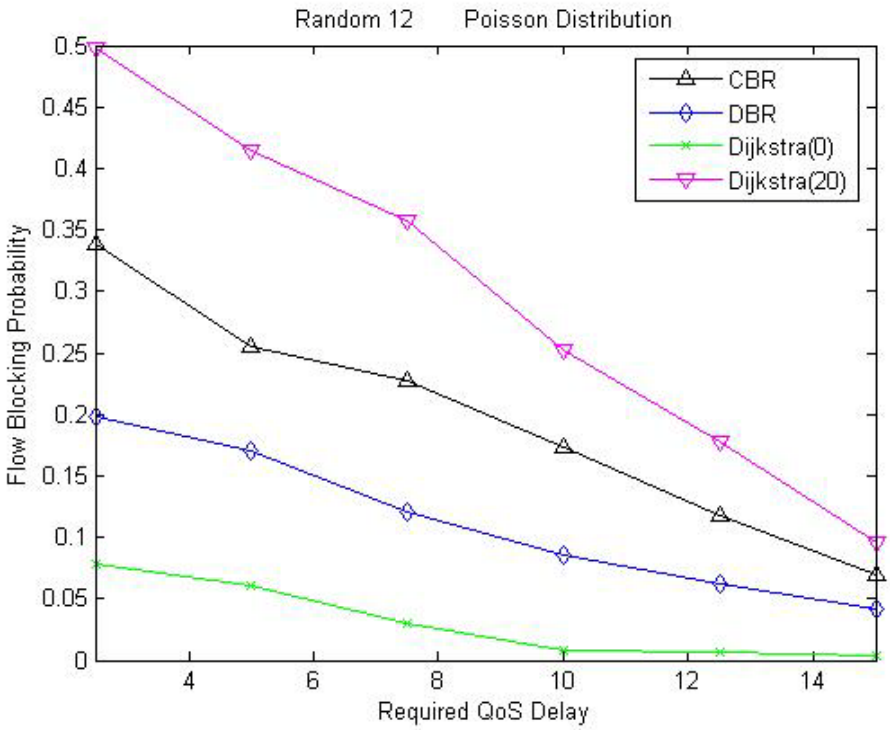
d Torus topology with Poisson distribution

Figure 6.4 Impacts of network topology

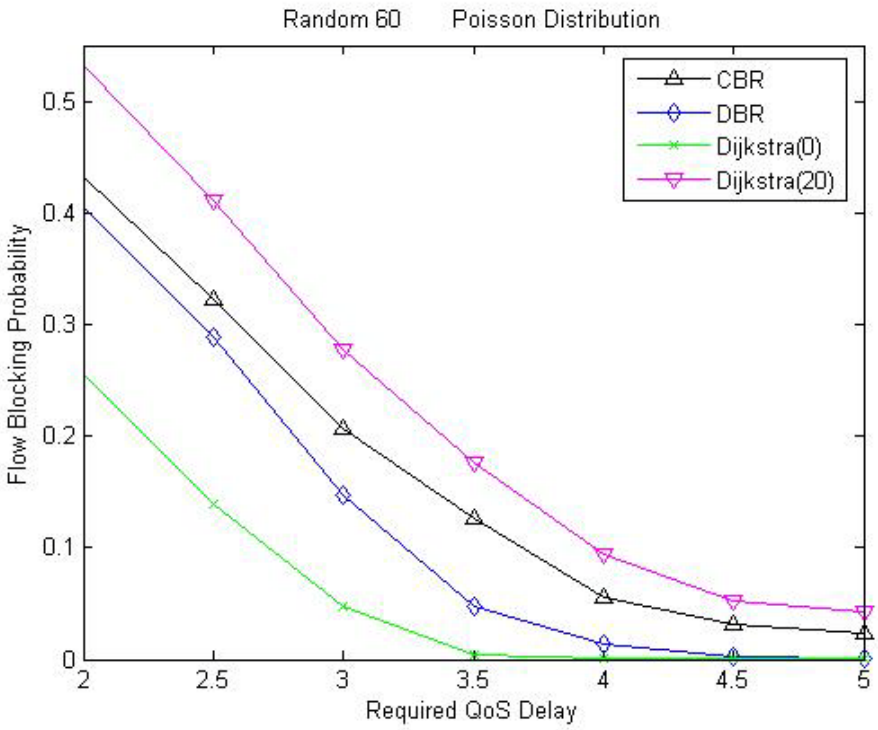
From Figure 6.4 we can assess the impact of the different network topologies. In all cases the DBR scheme performs better than the CBR scheme, Dijkstra(0) has the best performance, and Dijkstra(20) offers the worst performance. The difference between the performance of DBR and the performance of CBR is obvious. In the case of the Random 12 topology, when load increases, the difference becomes bigger and bigger. However in the cases of Random 60, ISP and Torus topologies, the difference becomes gradually closer and closer when load increases. In the Torus topology, all of these schemes give lower flow blocking probabilities than other topologies since the traffic is more uniform due to the regular topology. Through the comparisons we conclude that DBR has good performance for types of network topologies compared with the other schemes (discounting the unrealizable Dijkstra(0)).

6.4.4.3 Impact of QoS constraint

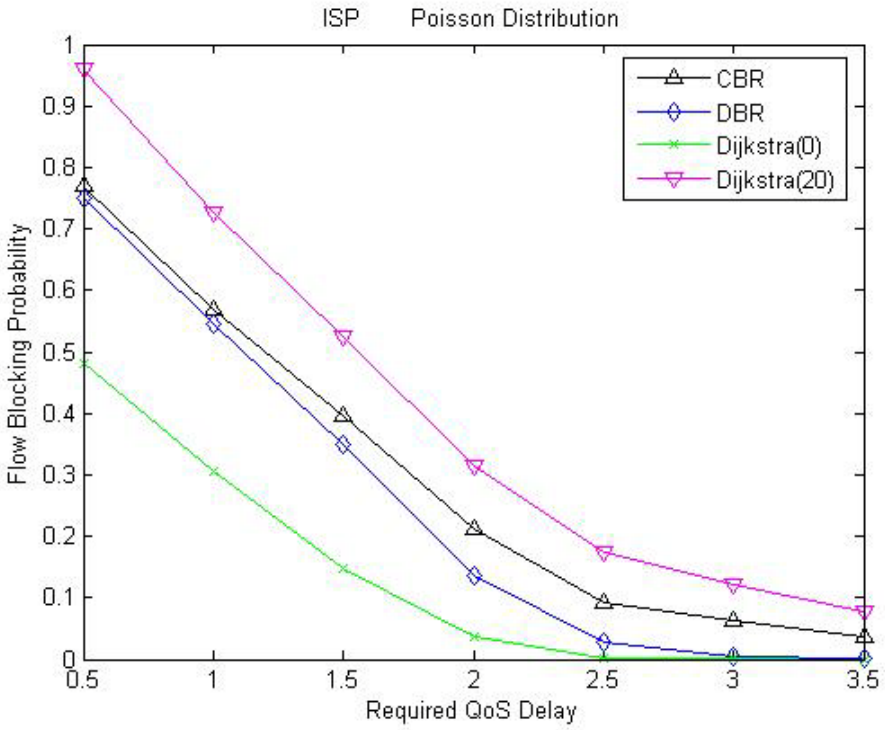
We have compared performance with different underlying network topologies against increasing load. Now we also compare them against increasing QoS constraint. In our simulation, we keep the load fixed as 0.4, where load 1 is set as the flow arrival intervals exponential(0.0005) seconds; then we get results as in Figure 6.5, which plot the blocking probability for the different algorithms against the required QoS delay, where the required QoS delay 1 is set as 1 second.



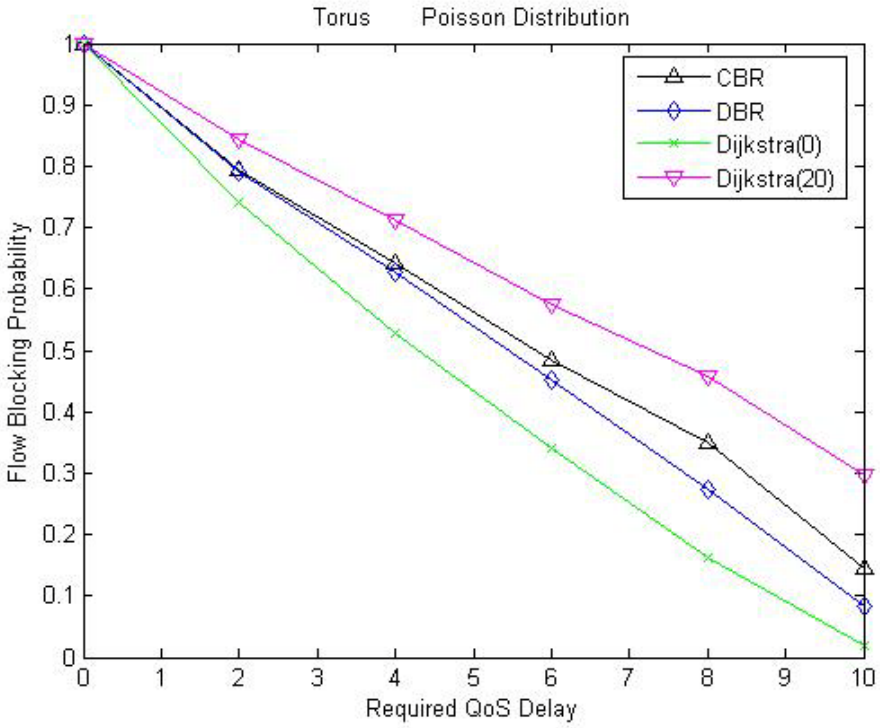
a Random topology RAND12



b Random topology RAND60



c ISP topology



d Torus topology

Figure 6.5 Impact of QoS constraint

Figure 6.5 shows while the required QoS delay increases, the blocking probabilities of different algorithms all decrease. This demonstrates that as the required QoS delay is increased, it becomes easier to satisfy the requirements since eventually all the available paths will satisfy the requirements if the required QoS delay is made large enough. In all cases the DBR scheme performs better than the CBR scheme, Dijkstra(0) has the best performance, and Dijkstra(20) offers the worst performance. In the case of the RAND12 topology, when the required QoS delay is small, the difference among all of algorithms is obvious; when required QoS delay increases, the blocking probabilities of all of algorithms are close stage by stage. However in the cases of RAND 60, ISP and Torus topologies, the difference becomes gradually more and more while required QoS delay increases. Through the comparisons we conclude that DBR has good performance in the different types of network topologies against increasing required QoS delay.

6.4.4.4 Impact of Bursty Traffic

Certain network traffic, such as IP traffic over Ethernets, has non-Poisson probability distributions and may actually show self-similar behaviour. It is therefore important that the flow arrival is bursty and distributions have heavy tails to model such traffic [118][119]. So although Poisson traffic is widely used to model network flow arrivals, we also model bursty traffic in the Random 60 topology to evaluate the performance of our algorithms. To model the burstiness of traffic, we use a Weibull distribution with two different values of the shape parameter of the distribution, 0.3 and 0.7, where burstiness is increased with a smaller shape value [116][119]. Figure 6.6 shows the flow rejection probabilities plotted against the offered load from 0.1 to 0.5 with different shape values.

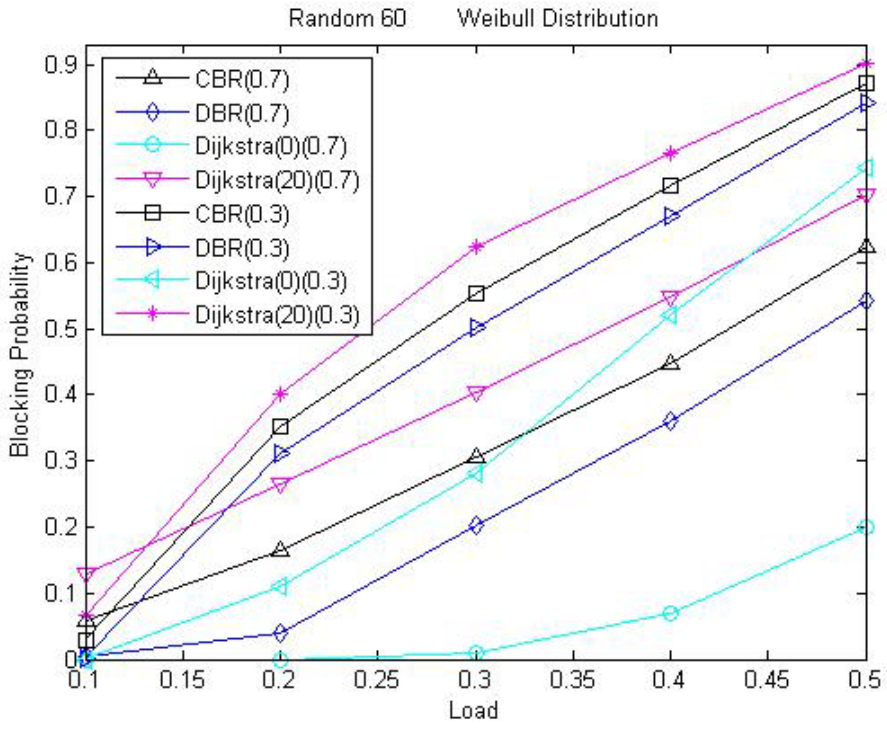


Figure 6.6 Impacts of bursty traffic

From Figure 6.6 we note that the more bustiness in the network arrival, the higher the blocking probability with the Weibull distribution. As with the Poisson distribution, the performance of Dijkstra(0) is the best Dijkstra(20) is the worst. The performance of DBR is better than CBR in the case of both shape 0.3 and 0.7, so DBR also has good performance with bursty connections.

6.4.4.5 Fairness

We also compare the fairness to evaluate performance of our proposed algorithm. Having a fair share of a resource is important where the resource demands of multiple flows sharing the resource are not met. In the absence of any other resource controls in the network, this means that there is at least one point along an end-to-end path where congestion is occurring, and we may determine how the resource is being shared by evaluating the resource distribution across the flows on that (part of) the path.

We again use Jain's Fairness Index (JFI) [104][105] to assess the system-wide fairness of CBR and DBR. The Jain's Fairness Index for the two schemes is shown as in Figure 6.7.

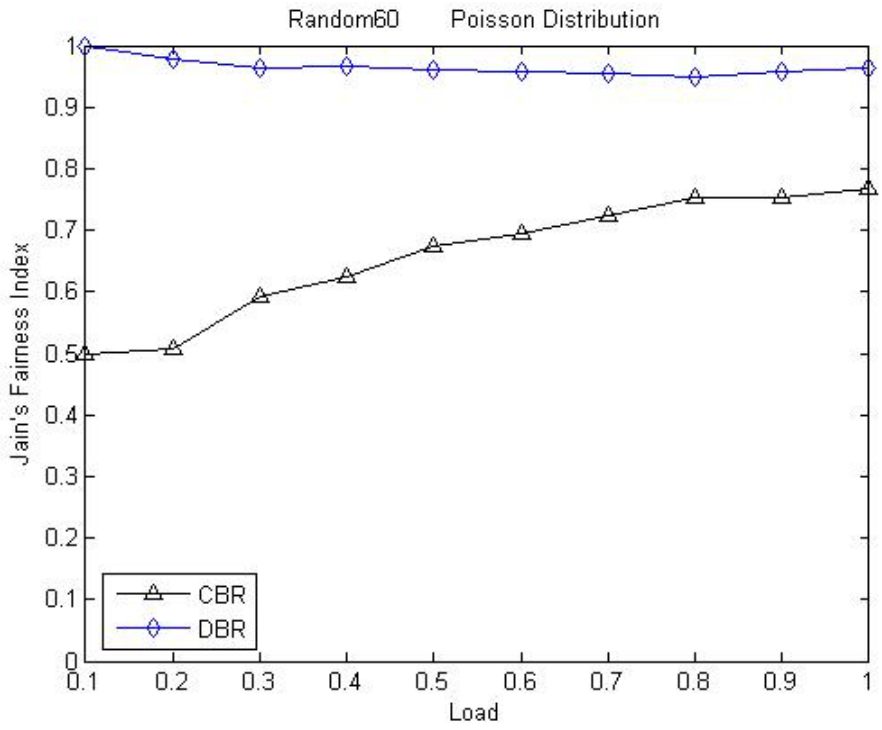


Figure 6.7 Jain's Fairness Index

Figure 6.7 shows as that: (1) DBR offers good JFIs which are all above 0.95; CBR gives bad JFIs that are between 0.5 and 0.77. DBR performance is better than CBR's. (2) JFI of CBR becomes high when loads increase. This demonstrates that CBR cannot reflect the actual current network QoS state, but only an outdated one, so it cannot share network resources fairly; DBR solves the problem and gets close to ideal fairness index, which is also relatively invariant across all loads tested.

6.4.4.6 Nodal Storage, Communication Overhead and Time Complexity

With regard to nodal storage, the candidate paths to each of the possible destination nodes would need to be stored at each node. However, it should be noted that the nodes would actually only be required to store the one hop neighbours for each of the

candidate paths and not the complete candidate paths. This would simply require the storage of the $R(N-1)$ addresses of the one hop neighbours of the candidate paths. A Connection Admission Control (CAC) feature can be built into DBR (as we have done in the simulation) by storing at each node for each existing flow through the node two variables representing the end to end required delay and the current end to end delay of the flows. The mean packet service time for any new flow can then be added to the current delay of any existing flow to check if this then exceeds its required delay and to react accordingly. The values of the required delay and the current delay of a flow can be stored at each node in a chosen path by the return message from the destination back to the source confirming that the path has been accepted. This would simply require each node to store, on average, $2F$ values, where F is the mean number of flows using the nodes. In addition, a node is required to store the chosen path for each of the flows originating from the node. Each path for a flow would be stored as an ordered set of nodes and would need to be carried in the header of each data packet belonging to the flow so that the packet can be routed accordingly.

With regard to time complexity, upon receiving a connection request, source nodes must first multicast a message to each of the one hop neighbours in the required candidate path set, requesting their current mean delays. Each of the neighbours then sends back a message to the source specifying the requested delays. The source node then compares the delays and selects the node with the lowest delay as the next hop. The source then sends a path set up message to this chosen node containing the source and destination addresses, the current cumulative end to end delay plus the link propagation delay, the required end to end delay, the required mean packet service time and the route chosen so far to the current node. Each intermediate node first checks to see if the required delay can be satisfied up to that node and that the required delays of

existing flows are not exceeded if the new flow is accepted. If the new flow is rejected then a corresponding message is returned to the source and any tentative resource reservations set up along the way are released. If the flow can be accepted up to the current intermediate node, this node then repeats the process carried out by the source node. The whole process is then repeated at each intermediate node until the next node is the destination. The destination then conveys the chosen route back to the source and confirms the path set up at each intermediate node on the return journey. Each intermediate node also must convey the new current end to end delay back to the source nodes of the flows through the node and all corresponding intermediate nodes update their current flow end to end delays. If the mean path length in hops, taken over all candidate paths, is H and k represents the other operations required at each node, such as checking the current mean delay through the node, then the mean number of operations in setting up a path is $H(Rk + 2F)$ Then as the network size grows without bound, the mean path length H will also increase without bound and, assuming R and F remain finite, time complexity is then O(H) and so is linear in H . Worst case time complexity is the same but with H the maximum path length. This is better than global algorithms but the same as CBR which, using mean delay as the QoS metric requires $R + H(2F + k)$ operations to set up a path. Thus, although the path selection is done entirely at a source node with CBR, the CAC and path set up still needs to be dealt with at each node in a candidate path and so time complexity remains at O(H) under the same assumptions as for DBR.

6.5. Summary

In this chapter we have introduced a delay based routing algorithm, DBR, which is a simple, localised QoS routing scheme which gives superior performance to the existing localised QoS schemes, PSR and CBR. These currently proposed localised QoS routing schemes operate with values that may be stale since they are only updated for a path when the path is used. In contrast to this, DBR uses current values of the QoS metric to make routing decisions and so is always operating with information that is likely to be more accurate and this undoubtedly contributes towards DBR's better performance. We have also used mean end to end delay as the QoS metric, which is a metric of considerable importance for real time services. This metric has not been considered in the previously published work on PSR and CBR, which have operated using residual bandwidth as the QoS metric. This latter metric does not require the use of CAC and is considerably easier to use than mean delay. However, DBR could easily be modified to use residual bandwidth by choosing the next node by comparing the reciprocal of the residual bandwidth on each outgoing link of the appropriate candidate path set and choosing the next hop with the lowest value. Time complexity for DBR with mean delay as the metric remains at O(H) since the algorithm has to be executed at each node in a path. With both PSR and CBR time complexity is still O(H), since the path set up involves operations at each node in the chosen candidate path. It should be noted that this was previously quoted as O(R) for both PSR and CBR but this considers only the path selection and excludes path set up, which is an integral part of the routing algorithm.

Through extensive simulations, we have compared the performance of DBR with that of the CBR scheme and demonstrated that DBR outperforms CBR in all cases examined. As a reference, we also compared the performance of DBR with Dijkstra's algorithm,

which uses global link state. The results indicated that DBR offers a better performance with much lower communication overhead than the Dijkstra scheme except in cases where Dijkstra has an unrealistically short update interval. Since DBR does not have to collect, update and distribute information relating to the global QoS network state, DBR also has much smaller nodal storage requirements and communication overhead compared to global schemes. Finally, the linear time complexity of DBR suggests good scalability properties.

- Same as bandwidth, delay can be an appropriate metric for a new localised QoS routing scheme which compares delays to make decision to select path. This scheme offers a good performance.
- In localised QoS routing, the schemes which explicitly reflect the quality of a path are better than schemes that are based indirectly on the path quality.
- Localised QoS routing can be employed to routes in a network with multi-constraint delay requirements or heterogeneous traffic.
- It is demonstrated that the proposed algorithms have good load balancing.

Chapter 7

Conclusions and Future Work

7.1. Conclusions

In order to meet the needs of real time and multimedia applications in modernized Internet applications, Qos routing is proposed to collect the state of link state information and based on this to find a feasible path that satisfies the QoS requirements. In current global QoS routing schemes, knowledge about the global network state is required. But with the expansion of modern networks and applications, to maintain an accurate global network QoS state is not practical. The unreasonable communication and processing overheads is caused by frequent exchange of QoS state information. Increasing update intervals to reduce routing overhead leads to inaccurate information of the global network QoS state due to stale resource information and this degrades the performance of QoS routing algorithms.

Localised QoS routing has been proposed to solve some of the problems in global QoS routing schemes. It infers the network QoS state in source nodes based on flow blocking statistics collected locally and performs flow routing using this localised view of the network QoS state. Various localised QoS routing algorithms have been proposed and it has been argued and demonstrated that they have advantages since the communication overhead and the processing and memory at core routers is greatly reduced.

This thesis has focused on localised QoS routing algorithms and has contributed a number of new insights into localised QoS routing as follows:

- We have developed three localised QoS routing algorithms (SBR, BBR and DBR). They have been compared with an existing localised QoS routing algorithm (CBR) and a global QoS routing algorithm (Dijkstra), and it has been shown that localised QoS routing can be considered a viable and scalable alternative approach to the use of global QoS routing algorithms.
- Unlike previous works which mostly use bandwidth as the required QoS metric, we have developed a novel localised QoS routing algorithm which selects a path using mean delay as a QoS single metric. We have argued that mean delay can be an effective metric to consider in QoS routing.
- In localised QoS routing, the schemes which explicitly reflect the quality of a path are better than schemes that are based indirectly on the path quality. This has been demonstrated in that our proposed three localised QoS routing algorithms outperform the existing localised QoS routing algorithm (CBR).
- It has been demonstrated throughout the thesis that localised QoS routing performs better than global schemes (with realistic update intervals) over different topologies and has better scalability properties and therefore might usefully be employed on the Internet.
- It also has been demonstrated that the three proposed localised QoS routing algorithms all have good load balancing properties.

7.2. Future Works

Possible directions for future works are listed as follows:

- Bandwidth and mean delay are used as QoS metrics in existing localised QoS routing algorithms research but jitter and packet loss probability have not been used. Therefore a study of localised QoS routing algorithms with jitter and packet loss probability as metrics might usefully be carried out in the future.
- The thesis has used bandwidth and mean delay as single QoS metrics to develop new algorithms. A multi-metric system which combines bandwidth and delay may be a good subject for future related work.
- QoS multicast routing has not been researched in the literature before. It has a lot of problems yet to be solved and must also be a frontier to be explored.
- In current networks, QoS mechanisms are based on the global QoS routing algorithms. If the QoS mechanisms attempt to change to localised QoS routing algorithms, the cost of network operations may initially increase and some existing applications will be affected. It would therefore be useful to research the transition from the global QoS routing algorithms to localised QoS routing algorithms in order to provide schemes which are non-intrusive and would not unduly disrupt existing services.
- To deploy localised QoS routing in the Internet would require localised schemes to be applied both within the subnetworks and also the backbone. A study of how this might be best achieved would also be an important contribution.

References

[1] A. Shaikh, J. Rexford, and K. Shin, “Evaluating the impact of stale link state on quality-of-service routing,” IEEE Trans. Networking, vol. 9, pp. 162–176, Apr. 2001.

[2] S. Nelakuditi, Z.L. Zhang, R. Tsang, and D. Du: “Adaptive proportional routing: a Localised qos routing approach”, IEEE/ACM Trans. Netw., 2002, 10, (6), pp. 790–804.

[3] S.H. Alabbad and M.E. Woodward, “Localised credit based QoS routing” Proceedings of IEE, vol.153, pp. 787-796, 2006.

[4] A. S. Tanenbaum, “Computer networks,” 4th ed. Upper Saddle River, N.J.: Prentice Hall PTR, pp. 353-354, 2003.

[5] George T. Heineman, Gary Pollice, and Stanley Selkow, “Chapter 6: Graph Algorithms” in Algorithms in a Nutshell., Oreilly Media, ISBN 978-0-596-51624-6, pp. 160–164, 2008.

[6] R. Coltun, D. Ferguson, J Moy, A. Lindem, “OSPF for IPv6,” in The Internet Society, RFC 5340, p.4, 2008.

[7] R. Atkinson, M. Fanto, “RIPv2 Cryptographic Authentication,” in The Internet Society, RFC 4822, pp.1-2, 2007.

[8] Muhammad Adeel, Ahmad Ali Iqbal, "TCP Congestion Window Optimization for CDMA2000 Packet Data Networks," itng, pp.31-35, International Conference on Information Technology (ITNG'07), 2007.

[9] Kurose, J.F, Ross, K.W, “Computer Networking: A Top-Down Approach Featuring the Internet,” Addison-Wesley, pp 1-6, pp 590-594, 2001.

[10] ETSI EG 202 009-1: “User Group; Quality of Telecom Services; Part 1: Methodology for identification of parameters relevant to the Users”. pp. 5-6, 2002.

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

[12] J. Evans, C. Filsfils, “Deploying IP and MPLS QoS for Multiservice Networks Theory and Practice,” Academic Press, pp.1-5, 2007.

[13] Desire Oulai, Steven Chamberland and Samuel Pierre, “An Improved Resource Reservation Protocol,” Journal of Computer Science 3 (8): 658-665, 2007.

[14] Sherif G. Aly, “Pervasive Differentiated Services: A QoS Model for Pervasive Systems,” World Academy of Science, Engineering and Technology , pp.1-6, 2005.

[15] Kodialam et al, “Dynamic Routing of Locally Restorable Bandwidth Guaranteed Tunnels using Aggregated Link Usage Information,” IEEE Infocom. pp. 376–385. 2006.

[16] A. Elwalid, C. Jin, S. Low, and I. Widjaja, “MATE: MPLS Adaptive Traffic Engineering,” in IEEE INFOCOM, pp. 1300–1309, 2001.

[17] A. Bouch, M.A. Sasse and H., DeMeer, “Of Packets and People: A User-centered Approach to Quality of Service”, IEEE IWQOS, pp.189 –197, 2003.

[18] Y. Koucheryavy, D. Moltchanov, and J. Harju, “A Top-down Approach to VoD Traffic Transmission Over DiffServ Domain Using AF PHB Class,” in Proc. IEEE ICC, Anchorage, AK, USA, pp. 243–249, 2003.

[19] B. Davie, A. Charny, J.C.R. Bennett, K. Benson, J.Y. Le Boudec, W. Courtney, S. Davari, V. Firoiu, D. Stiliadis, “An Expedited Forwarding PHB (Per-Hop Behavior), ” in IETF RFC 3246, pp.1-3, 2002.

[20] O. Younis and S. Fahmy, “Constraint-Based Routing in the Internet: Basic Principles and Recent Research,” IEEE Communications Surveys & Tutorals, vol.

5, pp. 2-13, 2003.

[21] M. Yuksel, K. K. Ramakrishnan, S. Kalyanaraman, J. D. Houle, R. Sadhvani, "IEEE International Workshop on Quality of Service (IWQoS'07)", pp. 109–112, 2007.

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

[23] V. Sharma, F. Hellstrand, “Framework for Multi-Protocol Label Switching (MPLS)-based Recovery,” in IETF RFC 3469, pp.1-2, 2003.

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

[25] P. Zhang and R. Kantola, “Building MPLS VPNs with QoS Routing Capabilitypp,” in Fifth International Symposium on Interworking, pp. 292-301, 2000.

[26] Jing Cao, Wei Wu, "A Multi-metric QoS Routing Method for Ad Hoc Network," msn, pp.99-102, 2008.

[27] ER.Yashpaul Singh, M.K.Soni, A.Swarup, “Throughput Analysis of SSP, SWP and MLS Routing Algorithms in Core Networks,” IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.8, pp.1-3, 2008.

[28] S. Alabbad, and M. E. Woodward, “Localized Credit Based QoS Routing: Performance Evaluation Using Simulation”, Proceedings of the 40th Annual Simulation Symposium, IEEE Huntsville, pp.1-2, 2006.

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

[30] N. Banerjee, S. K. Das, “MODeRN: multicast on-demand QoS-based routing in wireless networks,” Vehicular Technology Conference, pp.1-2, 2001.

[31] D. medhi, “QoS Routing Computation with Path Caching: A Framework and

Network Performance," IEEE Communications Magazine, vol.40, pp.106-113, December 2002.

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

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

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

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

[36] B. Lekovic and P. V. Mieghem, "Link state update policies for Quality of Serviec routing," in Eighth IEEE Symposium on Communications and Vehicular Technology in the Benelux (SCVT2001), pp.2-3, 2001.

[37] A. Stridgel and G. Mainimaran, "A Survey of QoS Multicasting Issues," IEEE Communications Magazine, vol.40, 2002.

[38] Sarosh Patel, Khaled Elleithy, and Syed S. Rizvi, "Hierarchically Segmented Routing (HSR) Protocol for MANET", 6th International Conference on Information Technology: New Generations ITNG 2009, Las Vegas, Nevada, USA, April 27-29, pp.1-7, 2009.

[39] D. Pei, D. Massey, and L. Zhang, "Detection of Invalid Routing Announcements in RIP Protocol", IEEE Global Communications Conference (Globecom), pp.1-6, 2003.

[40] D. Johnson, Y. Hu, D. Maltz, "The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4," in IETF RFC 4728, pp.4-5, 2007.

[41] G. Retvari, "Minimum Interference Routing: The Precomputation Perspective," Lecture Notes in Computer Science, 2003, Volume 2839, Management of Multimedia Networks and Services, pp. 246-258.

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

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

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

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

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

[47] D. S. Reeves and H. F. Salama, "A Distributed Algotithm for Delay-Constrained Unicast Routing," IEE/ACM Transactions on networking, pp.5-16, April 2000.

[48] Vinay Rishiwal, S. Verma and S. K. Bajpai, "QoS Based Power Aware Routing in MANETs," International Journal of Computer Theory and Engineering, Vol. 1, No. 1, pp.48-49, 2009.

[49] D. Ghosh, V. Sarangan, and R. Acharya, "Quality-of-Service Routing in IP Networks," IEEE Transactions on Multimediam, vol.3, pp.5-7, June 2001.

[50] Guozhen Tan, Yi Liu, Hengwei Yao, Jialin Li, Ningning Han, “A Hierarchical Routing Model for Large Scale Networks Based on Ant Algorithm,” International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICNICONSMCL'06), pp.88, 2006.

[51] Hongke Zhang, Stephan Olariu, Jiannong Cao, David B. Johnson, “Mobile Ad-hoc and Sensor Networks,” Third international conference, MSN 2007, Beijing, China, December 2007, Proceedings, pp.32-40.

[52] Hun-Jung Lim, Tai-Myoung Chung, “The Bias Routing Tree Avoiding Technique for Hierarchical Routing Protocol over 6LoWPAN,” 2009 Fifth International Joint Conference on INC, IMS and IDC, pp.232-235, 2009.

[53] A. Iwata and N. Fujita, “A Hierachical Multilayer QoS Routing System with Dynamic SLA Management,” IEEE Journal on Selected Areas in Communication, vol. 18, pp2603-2616, December 2000.

[54] 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.

[55] P. Paul and S. V. Raghavan, “Survey of qos routing,” in 15th International Conference on Computer Communication, Mumbai, India, pp.50-75, 2002.

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

[57] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, “Section 24.3: Dijkstra's algorithm,” Introduction to Algorithms (Second ed.). MIT Press and McGraw-Hill. pp. 595–601. ISBN 0-262-03293-7. 2001.

[58] George T. Heineman, Gary Pollice, and Stanley Selkow, “Chapter 6:Graph Algorithms,” Algorithms in a Nutshell. Oreilly Media. pp. 160–164. ISBN 978-0-596-51624-6, 2008.

[59] R.G. Dante, F. Padua, E. Moschim and J.F. Martins-Filho, “An Adaptive Routing Algorithm for Intelligent and Transparent Optical Networks,” Telecommunications and Networking - ICT 2004, Lecture Notes in Computer Science, Volume 3124/2004, 336-341, DOI: 10.1007/978-3-540-27824-5_46, pp.336-341, 2004.

[60] Mun Choon Chan, Yow-Jian Lin, “Behaviors and Effectiveness of Rerouting: a Study,” Communications, ICC 2005. 2005 IEEE International Conference, pp.1-6, 2005.

[61] J. Wang and K. Nahrstedt, “Hop-by-Hop Routing Algorithms For Premium-class Traffic In DiffServ Networks,” in Infocom, vol.2, pp.4-6, 2002.

[62] K. M. Elsayed, “HCASP: A hop-constrained adaptive shortest-path algorithm for routing bandwidth-guaranteed tunnels in MPLS networks,” Computers and Communications, Proceedings, ISCC 2004, Ninth International Symposium, pp.1-6.

[63] Vipin Kumar, Marina L. Gavrilova, Chih Jeng Kenneth Tan, “Computational Science and Its Applications,” International Conference, Montreal, Canada, May 2003, Proceedings, p.136.

[64] E. Atteo, S. Avallone, S.P. Romano, “A link weight assignment algorithm for traffic-engineered networks, ” Computer Networks 50 (2006) , pp.2286–2294.

[65] J.A. Khan, H. Alnuweiri, “A Traffic Engineered Routing Algorithm Based on Fuzzy Logic,” Communications, Computers and signal Processing, 2003. PACRIM. 2003 IEEE Pacific Rim Conference on, pp.454-457.

[66] S. Nelakuditi, “Localised approach to providing quality-of-service,” in Dept. of Computer Science and Engineering Minneapolis: University of Minnesota, p.174,

2001.

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

[68] 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.

[69] Deepankar Medhi, Karthikeyan Ramasamy, “Network Routing: Algorithms, Protocols, and Architectures,” Morgan Kaufmann Publishers, an imprint of Elsevier, pp.333-334, 2007.

[70] S. Nelakuditi, S.Varadarajan, Zhi-Li Zhang, On localized control in QoS routing, Automatic Control, IEEE Transactions on, pp.1-7, 2002.

[71] Deepankar Sharma, Ruchi Rani Garg, Amit Kumar Gupta, “Performance Measures of a Computer Network System,” Journal of Theoretical and Applied Information Technology, pp.187-202, 2009.

[72] V. Hulpic, “Simulation software: an operational research society survey of academic and industrial users,” in Simulation Conference Proceedings, 2000, Winter, 2000, pp. 1676-1683 vol.2.

[73] A. R. M. Shariff, “Some new distributed routing models and algorithms for Quality of Service routing,” Computer Communication Networks, 2007

[74] J. M. Pitts and J. A. Schormans, “Introduction to IP and ATM design and performance with applications analysis software,” John Wiley & Sons, 2002.

[75] C. Casetti, R. Lo Cigno, M. Mellia, M. Munafo, and Z. Zsoka, “A realistic model to evaluate Routing algorithms in the Internet,” in Global Telecommunications Conference, 2001. GLOBECOM ’O1. IEEE, 2001, pp. 1882-1885 vol.3.

[76] Paul Caspi, Jean-Louis Colaço, Léonard Gérard, Marc Pouzet, and Pascal Raymond,

"Synchronous Objects with Scheduling Policies: Introducing safe shared memory in Lustre," In ACM International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES), Dublin, pp.1-10, 2009.

[77] S. Floyd, M. Allman, "Specifying New Congestion Control Algorithms," in IETF RFC 5033, pp.1-10, 2007.

[78] P. Gevros, J. Crowcroft, P. Kirstein, and S. Bhatti, "Congestion control mechanisms and the best effort service model," Network, IEEE, vol. 15, pp. 16-26, 2001

[79] Xiumei Fan, Chuang Lin, Xiuping Fan, Yaya Wei, "Adaptive IP QoS Architecture and Algorithms," 2003 International Conference on Computer Networks and Mobile Computing (ICCNMC'03), p.107, 2003.

[80] Li Lao, Swapna S. Gokhale and Jun-Hong Cui, "Distributed QoS Routing for Backbone Overlay Networks," Lecture Notes in Computer Science, 2006, Volume 3976/2006, 1014-1025, DOI: 10.1007/11753810_84.

[81] F. A. Kuipers and P. Van Mieghem, "The impact of correlated link weights on QoS routing," in INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer Communications Societies. IEEE, 2003, pp. 1425-1434, vol. 2.

[82] Andy D. Pimentel, Stamatis Vassiliadis, "Computer Systems: Architectures, Modeling, and Simulation," Third and fourth International Workshops, SAMOS 2003 and SAMOS 2004, Samos, Greece, July 2003 and July 2004, Proceedings, pp.1015-1020.

[83] M. Klinkowski, D. Careglio, X. Masip-Bruin, S. Spadaro, S. Sanchez-Lopez, J. Sole-Pareta, "A Simulation Study of Combined Routing and Contention Resolution Algorithms in Connection-Oriented OPS Network Scenario," in *ICTON 2004*, Wroclaw, Poland, pp.33-36, 2004.

[84] Kevin Fall and Kannan Varadhan, "The ns manual", *URL: http://nsnam.isi.edu/nsnam/index.php/Main_Page*, May 9, 2010.

[85] "OPNET users' manual, OPNET architecture," OV.415. *http://forums.opnet.com*.

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

[87] A. Varga, "OMNeT++ Discrete Event Simulation System Version 3.0 User Manual", *URL: http://www.omnetpp.org*, Last updated: December 16, 2004.

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

[89] F. A. Kuipers and P. Van Mieghem, "The impact of correlated link weight on QoS routing," in INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE, 2003, pp.1425-1434 vol. 2.

[90] R. Pastos-Satorras and A. Vespignani, "Evolution and structure of the Internet," Cambridge: University Press, cambridge, pp.36-68, 2004.

[91] P. Erdos and A. Renyi, "On random graph," Publ. Math, pp.290-297, 1959.

[92] E. W. Zegura, K. L. Calvert, and M. J. Donahoo, "A quantitative comparison of graph-based models for Internet topology," IEEE/ACM Trans, Netw., vol.5, pp.770-783,1997

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

[94] M. Doar and I. Leslie, "How bad is naive multicast routing?" IEEE INFOCOM pp.82-89,1993.

[95] M. B. Doar, "A better model for generating test networks," in Global Telecommunications Conference, 1996. GLOBECOM '96, Communications: The Key to Global Prosperity, 1996, pp.86-93.

[96] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, “Measuring ISP Topologies with rocketruel,” Networking IEEE/ACM Transsctions on, vol. 12, pp. 2-16, 2004.

[97] O. Heekmann, M. Piringer, J. Schmitt, and R. Steimetz, “On realistic network topologies for simulation,” in Proseedings of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research Karlsruhe, Germany: ACM, 2003.

[98] D. E. Comer, “Internetworking with TCP/IP: Principles, Protocols and Architecture,” Prentice Hall, pp.13-30, 2000.

[99] S. Chen, and K. Nahrstedt: “Distributed quality-of-service routing in highspeed networks based on selective probing”. 23rd Annual Conf. on Local Area Networks, 1998, IEEE, pp. 80–90.

[100] C. Shigang and K. Nahrstedt, “On finding multi-constrained paths,” in Communications, 1998. ICC 98. Conference Record. 1998 IEEE International Conference on, 1998, pp. 874-879 vol. 2.

[101] G. Apostolopoulos, R. Guerin, and S. Kamat, “Implementation and performance measurnments of QoS routing extensions to OSPF,” in Processdings of IEEE INFOCOM, New York, 1999.

[102] S. Chen and K. Nahrsted, “On finding multi-constrained path,” in IEEE ICC’98, pp.874-899,1998.

[103] A. Medina, A. Lakhina, I. Matta, and J. Byers: “Brite: an approach to universal topology generation”. Proc. Int. Workshop on Modeling Analysis and Simulation of Computer and Telecommunications Systems, Cincinnati, Ohio, 2001, pp. 346–356.

[104] D.-M. Chiu and R. Jain. “Analysis of the increase and decrease algorithms for congestion avoidance in computer networks.” Computer Networks and ISDN Systems, 17(1):1–14, 1989.

[105] R. Jain, A. Durresi, and G. Babic. “Throughput fairness index: An explaination.” In ATM Forum/99-0045, 1999.

[106] R. Jain, D. Chiu, and W. Hawe, “A quantitative measure of fairness and discrimination for resource allocation in shared computer systems,” DEC Research Report TR-301, pp.3-8, 1984.

[107] A. M. law and W. D. Kelton, “Simulation modeling and analysis,” 3rd ed. New York: McGraw-Hill, pp.697-698, 2000.

[108] Z. Zhang, C. Sanchez, B. Salkewicz, and E. Crawley, “Quality of service extensions to OSPF”, Work in Progress, Internet Draft, Sept. 1997.

[109] H.D. Neve, and P.V. Mieghem: “TAMCRA: a tunable accuracy multiple constraints routing algorithm”, Comput. Commun., 2000, 23, (7), pp. 667–679.

[110] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick, “A framework for QoS-based routing in the Internet,” RFC 2386, pp.15-21, Aug. 1998.

[111] G. Apostolopoulos, R. Guerin, S. Kamat, and S. Tripathi, “Quality of service based routing: A performance perspective,” in Proc. ACM SIGCOMM 1998, Vancouver, Canada, Sept. 1998, pp. 17–28.

[112] Z. Wang and J. Crowcroft, “Quality-of-service routing for supporting multimedia applications,” IEEE J. Select. Areas Commun., vol. 14, pp. 1228–1234, Sept. 1996.

[113] Q. Ma and P. Steenkiste, “On path selection for traffic with bandwidth guarantees,” in Proc. IEEE ICNP 1997, Atlanta, GA, Oct. 1997, pp. 191–202.

[114] R. Guerin, S. Kamat, A. Orda, T. Przygienda, and D. Williams, “QoS routing mechanisms and OSPF extensions,” RFC 2676, pp.7-15, Aug. 1999.

[115] E.W. Dijkstra, “A note on two problems in connection with graphs”, Numerische Mathematick, pp. 1:269-271,1959.

[116] A. Shaikh, J. Rexford, and K. Shin, “Evaluating the impact of stale link state on quality-of-service routing”. Vol. 9, pp: 162 – 176, 2001

[117] A. Shaikh, J. Rexford, and K. Shin: “Efficient precomputation of quality of service routes”. Int. Workshop on Network and OS Support for Digital Audio and Video 1998, Cambridge, IEEE, 1998, pp. 15–27

[118] A. Feldmann, “Impact of non-poisson arrival sequences for call admission algorithm with and without delay,” in GLOBECOM’96, IEEE Global Telecommunications conference, Londeon, UK, pp.617-622,1996

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