Quality of Service Constrained Routing

By

Zeenat A. Khan

A thesis submitted to the

School of Computing

in conformity with the requirement for

the degree of Master of Science

Queen's University

Kingston, Ontario, Canada

July 2003

Your file Votre référence
ISBN: 0-612-86127-9
Our file Notre référence
ISBN: 0-612-86127-9

بسم الله الرحمن الرحيم

In the name of Almighty Allah,

The most Gracious, The most Merciful

Abstract

Multimedia applications, involving real-time audio/video teleconferencing and telemedicine require strict quality of service (QoS) constraints. Quality of service can be estimated and specified in terms of constraints or metrics that are of prime interest to the application such as end-to-end delay bound, bandwidth availability and loss probability. Substantial amount of work has been done on developing end-to-end QoS routing algorithms and protocols both for unicast and multicast flows for a wide variety of essential metrics and their combinations. To guarantee the real-time delivery of packets satisfying such constraints, QoS channel needs to be established in advance using a path selection algorithm that takes into account the QoS constraints. Establishing a connection that provides a guaranteed service involves routing, signaling, call admission, and resource reservation. A number of schemes that have been studied in this research work such as Delay-Constraint Unicast Routing (DCUR), Delay-Constraint Routing (DCR), and Distributed Delay-Constraint Algorithm (DDCA) are mainly focusing on the routing aspect of the problem while leaving the call admission and resource reservation problems for future investigation. In our work, we are analyzing and comparing the above- mentioned Delay-Constrained Least-Cost (DCLC) algorithms while integrating them with call admission and resource reservation in order to find the most viable path finding algorithm among the chosen schemes. We call this the Integrated Routing Protocol (IRP).

IRP establishes a unicast connection in two stages, a forward routing stage and a backward setup configuration stage. During the forward routing process, routing information is forwarded from a source node towards the destination on feasible paths through connection setup messages. The setup messages include the latest call admission control (CAC) information on links within the traversed paths. The destination node collects information on feasible paths through the above-mentioned routing algorithms in its database. Then it selects a viable least-cost path with the most available bandwidth that meets the delay constraint. The backward setup process starts after the destination node chooses the viable path and attempts resource allocation backwards towards the source node, such that the CAC criteria are met. The overall performance is enhanced as routing, CAC and resource reservation are integrated. A comprehensive simulation model is developed to study the performance of the proposed scheme. A number of experiments are conducted under different traffic characteristics and network parameters. Performance results show that IRP amplifies the probability of call acceptance by providing multiple-path choices between single pair of source and destination nodes. IRP is also shown to outperform existing DCLC routing algorithms in achieving low call blocking ratios as it adapts better to changes in the network and link characteristics.

Acknowledgement

I am extremely thankful to God, Who gave me the courage, strength and health to complete my studies. Deepest thanks and gratitude are extended to Professor Dr. Hossam Hassanein, for his untiring encouragement, kind support and patient discussions that were instrumental in the completion of this thesis. Heartfelt thanks are extended to Ali Roumani and Dr. Baoxian Zhou for providing all the help and guidance in due course of my studies. I would also like to thank all my friends in the Telecommunications and Networking Research group for being supportive and creating pleasant and friendly environment. Acknowledging the financial support provided by the School of Computing and the Natural Science and Engineering Research Council of Canada (NSERC) cannot be overlooked. Thanks would not be complete if not meant for Debby Robertson, Irene LaFleche, Tom Bradshaw and Richard Linley for their help.

I am extremely thankful to my ever-loving family for their moral support, long telephone calls, and prayers. I am also highly thankful to my husband Aman Khan for his wit and boundless support throughout this period that kept me going. I dedicate this work to him with many thanks.

List of Acronyms

ACK	Acknowledgement
CAC	Call Admission Control
Call_Setup_Req	Call Setup Request
CAR	Call Acceptance Ratio
CB	Cost Bound
CBR	Call Blocking Ratio
CBRDB	Call Blocking Ratio due to Delay Bound
CBRLB	Call Blocking Ratio due to Lack of Bandwidth
CBRSTE	Call Blocking Ratio due to Setup Time Expiry
DB	Delay Bound
DCLC	Delay-Constrained Least-Cost
DCR	Delay-Constrained Routing
DCUR	Delay-Constrained Unicast Routing
DDCA	Distributed Delay-Constrained Algorithm
FRBS	Forward Routing and Backward Setup
lc_nhop	least-cost next hop
ID	Identification
ld_nhop	least-delay next hop
IP	Internet Protocol
IRP	Integrated Routing Protocol
LC	Least Cost
LD	Least Delay
NACK	Negative Acknowledgement
QoS	Quality of Service
Req_Band	Required Bandwidth
RFC	Request for Comments
RR	Resource Reservation

Table of Contents

Table of Figures

Chapter 1
Introduction

IP has two primary responsibilities: providing connectionless delivery of packets through an internetwork (without any guarantee for an assured delivery of packets) and providing fragmentation and reassembly of datagram but without any guarantee of orderly arrival of packets at the destination. Such type of Internet routing traffic model provides Best Effort services that are not suitable for multimedia applications. The Best Effort connectionless service is a simple and scalable service that finds the shortest path from source to destination regardless of the constraints required by the applications such as bandwidth availability, delay, delay variations, buffer space, etc. The Best Effort model treats all the users or packets equally. Such a service would forward or discard packets solely based on Internet traffic conditions. If the network is over congested, and cannot forward packets then packets are simply dropped. Best Effort service is not good enough for newly emerging real-time multi-media applications such as video conferencing, Internet telephony, telemedicine, HDTV or video on demand, etc., where packet loss, low bandwidth, delay jitters are not tolerated. Therefore, there is a need for Internet architectures and protocols, including routing protocols that can provide an assured service to fulfill the required constraints of the multi-media applications independent of the network traffic condition. The Internet community referred to such type of services as Quality of Services (QoS). QoS routing is a set of routing mechanisms under which flow path selection is based on knowledge of network resources availability as well as flow QoS requirements. This thesis investigate various QoS routing schemes and proposes a novel approach of Integrated Routing Protocol to meet multi-media requirements and achieve efficient network resource utilization.

The remainder of this chapter is organized as follows. Section 1.1 describes QoS requirements for IP routing. Section 1.2 outlines the fundamental network functions such as, QoS constrained-base routing, call admission, resource reservation and interweaving all the three functions to achieve high network performance in terms of satisfying end-to-end QoS requirements of an application. Section 1.3 outlines the approach used in this thesis. Section 1.4 describes statement of problems and our two-fold research goal. Section 1.5 outlines the performance measures studied in this work. Section 1.6 presents an overview and structure of the remainder of the thesis.

1.1 QoS Requirements for IP Routing

It is critical for routing protocols to deliver data packets efficiently between a pair of source and destination nodes. The basic function of QoS routing is to find a network path that satisfies a set of given constraints of a connection. There are various QoS requirements, needed to be satisfied in order to provide assured delivery in IP routing. Such requirements include establishment of connection between end users, if the subnet is connectionless, whereas the routing decision must be made in every node visited and every packet routed. Therefore, IP-constrained routing requires connection establishment between end users within a well-defined call setup time limit. Selection of paths is the responsibility of QoS routing algorithms that are required to meet end-to-end delay bounds and available bandwidth, such that to optimize the network resources and manage network traffic in an efficient manner to enhance the network throughput.

1.2 Fundamental Network Functions

This section addresses fundamental network functions, such as QoS constraint routing, Call Admission Control (CAC) and resource reservation (RR). These functions must work closely with each other in order to provide guaranteed quality of service to applications. Our research interest lies in integrating all three closely related functions.

1.2.1 QoS Constrained-Base Routing

Routing algorithms determine a path from a source to a destination for the traffic flow. The routing algorithms used in current Internet routing protocols are mainly based on shortest path algorithms, while shortest path algorithm can only find a least delay path or find a path with most available resources [RFC1583]. Many heuristic solutions are proposed to address this problem. The QoS requirement of a connection is given as a set of constraints. For instance, a bandwidth constraint of a unicast flow requires that a feasible path between end systems have sufficient residual (unreserved) bandwidth, to satisfy the QoS constraints of a connection. Similarly, a delay constraint requires that the longest end-to-end delay between end systems not exceed an upper bound on the delay. The problem of QoS routing in distributed applications is difficult due to a number of reasons, as these applications require diverse QoS constraints on delay, delay jitter, loss ratio, bandwidth, etc. Our research objective is to study distributed delay-constrained QoS routing algorithms.

1.2.2 Call Admission Control

In order to maintain network load at a manageable level and guarantee QoS requirements of applications, we need an admission policy that grants or denies resource reservation before routing data packets over the Internet. We have to take into account the total resource allocation for a flow along a path in relation to available resources. If this flow needs too many resources we may reject it even if the network has enough capacity. This ensures that resources could be utilized by other flows with lower resource requirements.

1.2.3 Resource Reservation

Resource reservation usually follows routing. To begin communication, we must first find, a feasible path between the sender and receiver that meet the specific requirements, the application has set. To determine whether the QoS requirements of a flow can be accommodated on a link, a router must be able to determine if sufficient resources are available on the link [RFC2386]. QoS-based routing and resource reservation are closely related in a sense that their objective is the same, but they both perform different functions. Resources cannot be reserved unless the routing protocol can find a suitable path [Auk96]. So we need efficient mechanisms that integrate both functionalities to achieve better network performance.

1.2.4 Interweaving Routing, Call Admission Control and Resource Reservation

During the routing process, an admission control function is invoked at each link on the route being examined, to decide whether or not the QoS requirements of a flow can be accommodated on a link. The decision must be based on the incoming calls traffic characteristics and resources availability for the new connection as well as existing connections. Admission control is of significance, as it is often desirable to reject a request even when a feasible path has been found, if admitting the request will lead to inefficient use of network resources. Failure to put in such protection can result in throughput degradation in case of overload [RFC2676]. The information about the already allocated and available resources during QoS path computation improves the chances of finding a best path. Our approach to the problem of unicast connection establishment is to interweave routing with CAC and RR.

This thesis investigates three heuristics proposed in literature and presents our new approach of integrating admission control, resource reservation with distributed QoS-constrained routing algorithms. The objective of the admission control function as well the routing algorithm is to accept as many new connections as possible while guaranteeing the QoS requirements for every existing call [Hwa93, HaA00, Rou00], in order to fulfill real-time multi-media applications requirements. The paths for the connection are selected and resources are allocated, based on the requested QoS and the available network resources.

1.3 Approach

Several heuristic solutions have been proposed to address the problem of QoS routing. The approach we use consists on minimizing one of the QoS parameters under a second parameter constraint, for instance, minimizing a path cost under a delay constraint.

We interweave routing with CAC and resource reservation using a concept similar to Forward-Routing and Backward-Setup (FRBS) mechanism [HaA00]. This research aims to find the most viable path between source and destination nodes in two stages. The first stage computes QoS paths using distributed delay-constrained unicast routing algorithms described in Delay-Constrained Unicast Routing (DCUR) proposed by Salama et al. [SRV97], Delay-Constrained Routing (DCR) proposed by Sun and Langendorfer, [SuL98] and Distributed Delay-Constrained Algorithm (DDCA) proposed by Zhang et al., [ZKM01]. Call admission control is used in routing to test if a path can support the call (connection)[1], while maintaining the QoS of the existing connections. In the second stage, the destination node selects the path with best costs value and begins resource reservation backwards towards the source node, exercising CAC on each link along the path. The routing algorithm searches for all the available paths to increase number of accepted connections (HaA00). CAC decisions are based on the current network traffic load.

The rationale behind choosing DCUR, DCR, and DDCA routing schemes for our research work is their common objective to solve the unicast delay-constrained cost minimization problem. The common properties these schemes exhibit are as follows:

- All these schemes work in a hop-by-hop fashion, using a distributed routing strategy.
- Each node knows the minimum cost and minimum delay to every other node in the network.
- The cost vector and delay vector at all nodes are assumed to be up-to-date.
- Distance-Vector or Link-State protocols are used to support the dynamic nature of Internet traffic.
- Periodic updates are sent only to intermediate neighbor nodes, instead of flooding the entire network. Only limited state information are needed at each node, requiring small amounts of computation.

[1] The term call and connection are used interchangeably.

- The path is constructed one node at a time, each time the added node lies on either the least-cost (LC) or least-delay (LD) path.
- Routing metric, path computation and algorithm complexity are the important common criteria taken into account for choosing the above mentioned routing schemes.

1.4 Statement of Problem and Research Goal

The need to provide a guaranteed QoS requires a call-level admission control mechanism and the reservation of resources on link-by-link basis. Providing QoS in IP network is complicated due to its connectionless nature. The need to reserve resources (e.g., bandwidth) for each individual connection in order to guarantee its QoS requirements has made connection establishment in high-speed network indispensable. This is the responsibility of the resource reservation and admission control mechanisms to provide information about residual resources (bandwidth) on a link such that path selection decisions are based on this information. A bandwidth constraint of a unicast connection requires, for instance, that the links composing the path must have certain amount of free bandwidth available [RFC2386] to assure real-time channel establishment. The establishment of such channels requires the development of efficient route selection algorithms that are designed to take into account the QoS constraints.

As mentioned above, this thesis aims to study and compare the Distributed Delay-Constrained Least-Cost (DCLC) unicast routing algorithms proposed in literature. The algorithms are analyzed and compared in an integrated setting with call admission and resource reservation, in order to find the most viable path finding algorithm among the chosen schemes. This approach is denoted as Integrated Routing Protocol.

Our destination-controlled Integrated Routing Protocol establishes a unicast connection in two stages, a forward routing stage and a backward setup stage. During the forward routing process, the routing information is forwarded from source towards destination on feasible paths through connection setup messages. The setup message includes the latest call admission control (CAC) information on links within the traversed paths. The destination node collects information on feasible paths through above-mention routing algorithms in its database. Then it selects a viable least-cost path with the most available bandwidth that meets the delay constraint. The backward setup process starts after the destination node chooses the most viable path and attempts resource allocation backwards towards the source node, such that the CAC criteria are met. The main objective of this effort is set as two fold:

- To enhance the developed routing strategies by analyzing and interweaving them together with call admission control as well as resources reservation.
- To introduce a new approach of Integrated Routing Protocol (IRP), that provides a mechanism for path computation that is superior to existing routing strategies.

1.5 Performance Measures

The network performance objective is to establish real-time channels that attempt to maximize the average call acceptance ratio by admitting calls with sufficient resources. IRP enhances the probability of call acceptance by providing multiple paths choices between single pair of source and destination node. The proposed IRP scheme finds loop-free routes that satisfy the requirements of individual flows and adapts well to changes in network and link characteristics as it achieves low call blocking ratios. The IRP is shown to achieve a high success ratio for call establishment, and hence results in enhancing the overall network performance.

1.6 Thesis Overview

The remainder of thesis is organized as follows. Chapter 2 presents a survey of QoS routing and related work such as routing principles, routing algorithms and routing protocols. The classification of routing protocols into distance-vector and link-state routing protocols and the QoS routing strategies such as source routing, hierarchical routing and distributed routing are presented with a brief literature review of various proposed algorithms under each strategy.

Chapter 3 presents a unicast routing problem formulation and network model description. A detailed description and evaluation of the Delay-Constrained Least-Cost routing algorithms such as Delay-Constraint Unicast Routing (DCUR) by Salama et al. [SRV97], Delay-Constraint Routing (DCR) by Sun and Langendorfer [SuL98], and Distributed Delay-Constraint Algorithm (DDCA) by Zhang et al., [ZKM01] is presented. Various aspects of the selected distributed delay-constrained unicast algorithms are addressed such as routing information needed for routing algorithms and connection establishment requirements in connectionless IP networks.

Chapter 4 proposes a model of the Integrated Routing Protocol (IRP) approach. It also describes the design and implementation of various algorithms and presents pseudo code for source node, intermediate and destination nodes both at forward routing stage as well as backward setup stage.

Chapter 5 presents the performance evaluation of the experimental setup. A comprehensive simulation model is developed to study the performance measures of the proposed scheme. A number of experiments are conducted under different traffic characteristics and network parameters. The performance metrics studied are the call blocking probabilities and call acceptance ratio.

Chapter 6 concludes the thesis with summary of the research contributions.

Chapter 2
Overview of QoS Routing and Related Work

The routing process consists of two fundamental stages at the network layer [Sal96]. The first stage consists of selecting a route for the session during the connection establishment phase, and the second stage consists of forwarding real data packets of that session along the selected path. In this research we focus on the first task only and assume that the true state of the network is available to every node via distance vector routing protocol such as Bellman Ford's shortest path algorithm. Network nodes use this information to determine end-to-end least cost as well as least delay paths. Each link in the network is associated with certain parameters that provide the measurable QoS metrics. The selection of routing metric is one of the key design issues that determine the criteria for path selection. In addition to this, routing metric has an important implication on the complexity of path computation [WaC95]. Well known metrics include bandwidth, delay, jitter, cost and loss probability.

The routing problem can be classified into two major classes: *unicast routing* and *multicast routing*. Unicast routing refers to finding a feasible path between a single source and a single destination. On the other hand, multicast routing refers to finding a feasible tree covering a single source or multiple sources and a set of destinations. Our research focus is on unicast routing algorithms. The routing algorithms are classified into static routing, adaptive routing and dynamic routing. In static routing, all routing decisions are known at network setup time and are fixed, independent of the network state. Dynamic routing allows routing decisions to vary over time, but not necessarily depends on the network state. Finally, in adaptive routing, the routing decisions are based on a function of some estimate of the network state and thus may also vary over time [Hwa93].

The rest of this chapter is structured as follows. Section 2.1 present routing principles and outlines a defined line of difference between routing algorithms and routing protocols. Section 2.2 presents classification of routing protocols into distance-vector and link-state routing protocols. Section 2.3 describes routing algorithms and their types. The classification of routing algorithms into various strategies is presented in Section 2.4. The well-known QoS routing strategies are source routing, hierarchical routing and distributed routing. However, recent work on QoS routing has been focusing on two main directions: source routing and distributed routing. In source routing, each node maintains an image of the global state of the network, based on which a routing path is centrally computed at the source. Whereas, in distributed routing, the path computation process is carried out in a distributed fashion by finding the next best hop in the path. Control messages are exchanged among the nodes and the state information collected at each node is used in order to find a path. A brief introduction along with a review of various proposed algorithms of the two strategies is presented in Section 2.4. Section 2.5 concludes the chapter with a discussion of QoS routing and related work in form of short summary.

2.1 Routing Principles

Routing is the main process used by Internet hosts to deliver packets, depending on routing algorithm and routing protocol. A routing protocol is a set of rules implemented at the network layer, selecting the least-cost path to the destination. A routing protocol forms the core of the Internet. At the core of any protocol is a routing algorithm that determines routes connecting a set of network nodes belonging to a particular session [TEL316]. A protocol should be robust and fault tolerant so that it reacts fast and safely to link or node failures in order to minimize the resulting instability in the network. Path selection within routing algorithms is formulated as a cost-optimization problem. The objective function for optimization could be any one of a variety of parameters, such as least end-to-end delay, hop count or bandwidth utilization. At the current stage of network evolution, it may be appropriate and important to design a simple, scalable routing algorithm that satisfies a given end-to-end delay bound and manages the network resources efficiently, particularly if the designed algorithm can be easily integrated into the current Internet routing protocols. Our objective is to study the performance of routing algorithms and not the routing protocols.

2.2 Classification of Routing Protocols

Routing protocols are used between routers and represent additional network traffic overhead on the network. An important feature of a routing protocol is its ability to sense and recover from failures. How quick it can recover is determined by: the type of fault, how it is sensed, and how the routing information is propagated through the Internetwork. When all the routers on the internetwork have the correct routing information in their routing tables, the internetwork has converged. When a link or router fails, the internetwork must reconfigure itself to reflect the new topology. Information in routing tables must be updated. Until the internetwork re-converges, it is in an unstable state with inconsistent information. Internet routing protocols can be classified into two categories: distance-vector protocol and link-state protocol. The following section describes these categories.

2.2.1 Distance-Vector Protocols

Distance-Vector routing protocols, such as Routing Information Protocol (RIP) are based on a distributed version of the Bellman-Ford Shortest Path (SP) algorithm. Each node maintains only limited information about the shortest path to all other nodes in the network. Upon receipt of an update, for each destination in its table a router compares the metric in its local table with the metric in the neighbor's node and also the cost of reaching that neighbor. If a path via neighbor has a lower cost, the router updates its local table to forward packets to the neighbor.

A problem with such protocol is that it may continue to use old information that is invalid, even after network topology changes and new information becomes available. Due to their distributed nature, distance-vector protocols may suffer from looping problems when the network is not in a steady state. On the plus side, and considering message complexity, distance-vector routing protocols scale well to large network sizes, because each node (router) sends periodical topology update messages only to its direct neighbors.

2.2.2 Link-State Protocols

Link-State protocols such as Open Shortest Path First protocol (OSPF) are based on Dijkstra's SP algorithm. A database is maintained at each node that describes network topology and link delays between each router. Each router keeps track of the complete graph of links and nodes in the network. Therefore, each router periodically discovers its neighbors and measures delays across its links then forwards this information to all other routers. Updated information is propagated at high priority using flooding technique. Updates contain sequence numbers and a router forwards "new" copies of the packet. This way routing updates propagate even if routing tables are not quite correct. Acknowledgments are sent to neighbors. Each router uses the Shortest Path First (Dijkstra's SP) algorithm to compute the shortest path based on the current values in its database. Since each router makes its calculation using the "same" information (depending on the update frequency), accurate routing decisions can be made. Link-state protocols also do not suffer from looping.

2.3 Routing Algorithms

Routing algorithms search for paths from a source to a destination. The path that appears to be most promising among all possible paths is selected. Routing decisions affect the network behavior for the duration of a call. Routing is particularly crucial to network performance. There are two types of routing algorithms currently in practice; adaptive and non-adaptive algorithms. Adaptive algorithms are flexiable and adapte well to changes in network topology, load, delay, etc., to select routes. On the other hand, non-adaptive algorithms are static and routes do not change. Adaptive algorithms can be further classified into following types: isolated, centralized and distributed:

- **Isolated:** Each router makes its routing decisions using only the local information it has on hand. Specifically, routers do not even exchange information with their neighbors.
- **Centralized:** A centralized node makes all routing decisions. Specifically, the centralized node has access to global information.
- **Distributed:** Algorithms that use a combination of local and global information. In a distributed routing mechanism a path is computed in a distributed manner on a hop-by-hop fashion.

In this work we opt to study the behavior of adaptive distributed routing algorithms.

2.4 Routing Strategies

Distributed multi-media applications have quality of service requirements specified in terms of constraints on various metrics such as bandwidth, delay, delay jitter, cost etc. Various routing strategies could be adopted to find the feasible path between source and destination nodes. There are three well-known routing strategies commonly in use. Their classification depends on how state information is maintained and how the search for feasible path is carried out [ChN98b]. These are: source routing, hierarchal routing and distributed routing.

2.4.1 Source Routing

In source routing, each node maintains an image of the global network state. The global network state refers to the information regarding the network connectivity and resource availability, based on which the entire routing path is computed at the source node. The link-state protocol is used to periodically update the network state. Finding a path in source routing can be computationally intensive for the source router. The overhead of the source routing algorithm lies in the fact that a huge amount of storage capacity is required at each router in the network to maintain global state information. The global state thus maintained is inherently imprecise due to the dynamic nature of network resources availability [GSA01]. There is always a tradeoff between the average number of messages exchanged and the amount of staleness or impreciseness in the global state maintained at each router. The amount of impreciseness and the average message overhead both increases with the network size. Hence, such approaches are not scalable to large network size [GSA01]. In the following we review various source routing algorithms in the literature.

A throughput competitive routing algorithm for bandwidth-constrained connection was proposed by Awerbuch et al. [AAP93]. The algorithm tries to maximize the average throughput of the network over time. It combines the function of admission control and routing. Every link is associated with the cost function that is exponential to the bandwidth utilization. A new connection is admitted into the network only if there is a path whose accumulated cost over the duration of the connection does not exceed the profit measured by the bandwidth-duration product of the connection. It was proved that such a path satisfies the bandwidth constraint. Let T be the maximum connection duration and n be the number of nodes in the network. The algorithm achieves a throughput that is $O\ (Log\ nT)$ factor of the highest possible throughput achieved by the best off-line algorithm that is assumed to know the entire connection request in advance.

Another heuristic that combines source routing with call admission was proposed by Liu and Mouftah [LiM95]. The proposed scheme is known as a Virtual Call Admission Control (VCAC) algorithm that examines a routing database of all links in the entire network before a path is selected thus resulting in an effective topology database. The database includes links that follow the CAC scheme and are able to accept calls. Thus chances of new call acceptance are enhanced along a path selected from the effective topology than from the entire topology. To avoid possible delays, link metrics used by CAC and routing in VCAC are advertised through link state updates and stored in the network topology database. The concept of effective topology and Actual Call Admission Control (ACAC) is used in the proposed VCAC scheme. Upon receiving the call setup request, the VCAC is first exercised on every link included in the entire topology. Dynamic routing protocols such as Shortest Path First (SPF) and Open Shortest Path First (OPSF) are used to select the path. Each link on the selected path is examined through its own ACAC. Then VCAC algorithm is performed upon every link in the entire topology to decide about the link to be included into the effective topology. This decision is based on the call characteristics and values of link metrics available in the database topology. The links that pass the VCAC check are likely to pass the ACAC check as well, resulting in a path construction on such an effective topology with high probability. Such links will be included in the topology to achieve high network performance. The proposed VCAC algorithm showed good performance, with its requirement of only one metric and its computational simplicity using any VCAC strategy.

Wang and Crowcroft [WaC96] investigated the routing problem subject to multiple quality of service constraints. They studied the multiplicative, concave and additive constraints, and proposed a well-known shortest widest path algorithm. This algorithm finds a bandwidth-delay-constrained path using Dijkstra's shortest path algorithm. All links with bandwidths less than the requirement are eliminated such that remaining paths in the resulting graph will satisfy the bandwidth constraint. Thus shortest path in terms of delay is found. The path is feasible, if and only if, it satisfies the delay constraints.

A novel forwarding technique for routing with multiple QoS constraints is proposed by Fei and Gerla [FeG00]. The proposed technique can be applied both in distributed hop-by-hop routing and source-based approach. The key idea of the smart forwarding is that when constructing an end-to-end path, one can make some "smart" decision on which next hop should be in the path to satisfy the constraints and minimize the cost. It can be used as a crank back approach in which it attempts another path if one does not work out. However, it tries a path only if it knows there is a chance that given path might be feasible. The algorithm performance and processing overhead can be adjusted by setting a limit on how many crank-back trials or forwarding branches it can have. It utilizes a per-computed table (updated periodically) that can be constructed with low computation complexity and each on-demand routing request can then be answered with low processing overhead. When used as a flooding scheme without limiting crank-back trials, it can greatly reduce the number of routing messages since a routing request will only be sent to promising neighbors based on "smart" decisions. Simulation results show it is always effective in finding low cost paths.

2.4.2 Distributed Routing

In distributed routing, path selection is carried out in a hop-by-hop fashion. The Distance-Vector protocol is used to periodically update the intermediate neighbor nodes. Every node maintains global state information only about its next neighbor in the form of a distance vector (tables), returning the best next hop only. Since the path computation process is shared among intermediate routers, there is no computational burden on any single router in the network [GSA01]. Hence the routing response time can be made shorter to enhance the network scalability and making the distributed routing mechanism more suitable for routing with dynamic nature of traffic conditions. However, it may suffer from looping problem if the global state information at nodes is not kept consistent. Loops will cause routing failure because distance-vectors do not provide sufficient information for an alternative path. In the following we review various distributed routing algorithms in the literature.

Distributed algorithms can be categorized into two types based on whether all the routers maintain a global state or not. If the routers have a global state, such information can be used in path computation to specify the best next hop. If no global state is stored then flooding techniques can be used to establish a path, where a request is flooded on all outgoing links that satisfy the QoS requirements of the request. The problem with such an approach however, is that the overhead involved in establishing a connection may be high.

Salama et al. [SRV97] proposed a distributed delay-constrained unicast routing (DCUR) algorithm. A cost vector and a delay vector are maintained at every node by a distance-vector protocol. A control message is sent from the source toward the destination to construct a delay-constrained path. Any node *i* at the end of the partially constructed path can select one of only two alternative outgoing links. One link (*i, j*) is on the least-cost (LC) path that is directed by the cost vector while other (*i, k*) is on the least-delay (LD) path directed by the delay vector. Link (*i, j*) has the priority to be chosen, as long as adding the least-delay path from *j* to the destination does not violate the delay constraint.

Reeves and Salama [ReS00] has proposed a revised version of DCUR after Salama et al. [SRV97], which simply ignores from further consideration any least-cost links, once loop removal has removed that link due to a loop occurrence. An entry in the invalid-link table at the corresponding node is created and saved. DCUR worst-case message complexity is $O(|V|^2)$.

Sun and Langendorfer [SuL98] proposed delay-constrained routing (DCR) and improved the worst-case performance of the DCUR algorithm by avoiding loops instead of detecting and removing loops. A control message is sent to construct the routing path. The message travels along the least delay path until it reaches a node from which the delay of least-cost path satisfies the delay constraint. From that node on, the message travels along the least cost path all the way to destination. The message complexity of the DCR algorithm is $O(|V|)$.

Sriram et al. [SMM98] proposed a preferred link distributed delay-constrained least-cost routing algorithm in order to remove the restriction on path selection method in DCUR as proposed by Salama et al. [SRV97] and Sun and Langendorfer [SuL98] algorithms. It provides better performance in terms of increased call acceptance rate and lower average route cost. It also combines resource reservation with probing thus avoiding a separate reservation phase. The preferred link distributed delay-constrained least-cost routing algorithm is fundamentally a backtracking-based route selection method. At each node a set of actions is performed whenever it receives a call setup or a call reject packet. When a node **v** receives a call setup packet, it forwards it along the first preferred link. If a reject packet is received from the node at the other end of this link, then node **v** attempts to forward the packet along the next preferred link and so on, until a specified number of links have been tried out. If all such attempts result in failure, then **v** sends back a reject packet to the node from which it received the call setup packets. If the call setup packet reaches the destination, then the call is successfully setup.

Zhang et al. [ZKM01] proposed a distributed delay-constrained algorithm (DDCA). DDCA is basically an extension of DCR proposed by Sun and Langendorfer [SuL98]. The path construction process is replaced by path probing process and extends the probing direction to include both the least-cost (LC) and least-delay (LD) direction. DDCA is based on a relay strategy. The relay node is a node that connects two super-edges (connected segment of the path on which all the routers use the same metric for packet forwarding) on a selected path. The minimum cost path returned by the proposed algorithm will always be loop free. In DDCA, each node in the network maintains a *delay table* and a *cost table*. The information in the delay and cost tables can be distributed to nodes using distance vector protocols. More details on DCR, DCUR and DDCA are provided in Chapter 3.

The Distributed Routing Algorithm (DRA) framework based on selective probing is another approach used for solving the DCLC problem and was proposed by Chen and Nahrstedt [ChN98a]. The DRA has three phases: probing, acknowledgment and failure handling. The probing phase is essentially the QoS routing and it establishes a *tentative path* between the source and destination such that the path satisfies the QoS requirements of the connection. A router on receiving a probe, selectively floods it on all the outgoing links, which are capable of supporting the QoS requirements, except the one on which the probe arrived. Every router selectively floods the first probe received for a given connection and rejects all duplicates. The probing phase ends once a probe reaches the destination, and the path that the first probe takes to reach the destination is called the *tentative path*. No reservations are made in the probing phase. The destination sends an acknowledgment to the first probe it receives and it discards all the duplicate probes. This ensures that only one path is established. In the acknowledgment phase, the destination sends an acknowledgment along the tentative path and resources are reserved along the way. A connection is established when the source gets the acknowledgment. As the network resources change rapidly, an intermediate node on the tentative path (which would have forwarded the probe earlier) might not have the required resources when it gets an ACK for the same connection. In such a case, the failure-handling phase is started. The node that was unable to reserve the resources sends a failure message to the downstream routers to free their resources and the connection request is refused. A router could receive a probe with certain QoS constraints which none of its outgoing links can support. The router will simply discard the probe without forwarding it. If such a condition occurs in all the paths between a source and destination pair, the connection will not be set up. In both cases, the call is blocked. The probing algorithm substantially reduces the routing overhead at the cost of long routing time. If every node maintains a global state, which is allowed to be imprecise, the ticket-based probing is used to improve the performance of selective probing [ChN98b]. A certain number of tickets are issued at the source according to the contention level of network resources. Each probe must contain at least one ticket in order to be valid. Hence, the maximum number of probes is bounded by the total number of tickets, which limits the maximum number of paths to be searched. The algorithm utilizes the imprecise state at intermediate nodes to guide the limited tickets along the best possible paths to the destination. In such a way, the probability of finding a feasible path is maximized with the limited probing overhead.

Enhancing the selective probing approach of Chen and Nahrstedt [ChN98a], two heuristics are proposed by Sarangan, et al. [SGA00] for selectively flooding the probes so that the overall resource admission rate is increased. The proposed strategy is for metrics like bandwidth, switch buffer, CPU usage etc., where the minimum value of the metric at each router along a path should satisfy the QoS constraint. Examples of metrics that do not come under this category are delay, jitter, etc. In the first proposed heuristic, probes travel through paths that fit the QoS request more precisely, reach the destination faster than probes that travel through other paths. The acknowledgment (ACK) would then be sent back along this path of close fit. The intuition behind this approach is to discourage fragmentation of available resources. If the ACK is sent back on the path of best fit, the chances of resources getting used up at an intermediate node in this path before the ACK reaches that node is quite high. Hence, it would seem that the call admission rate should decrease in this approach instead of increasing. However, experimental results showed increase in number of connections established due to this strategy outnumber the number of connections that are rejected. Another strategy to increase the call admission rate is that if a destination has many paths to choose from, it sends the ACK on the path that has the maximum resources. The probability of the resources getting used up before the ACK reaches an intermediate router on this path is lower when compared to a router on any other path. The idea behind the first approach is that discouraging resource fragmentation maximizes a number of connections, whereas in the second approach, the chances of generating a failure message along the tentative path are reduced. It is assumed that the control messages have a higher priority over the data packets. They will not be dropped in any case, even if the network gets congested and will be serviced immediately at the routers. Hence, the control messages incur a negligible queuing delay at all routers. The only delay they experience in a router is the protocol processing time. Furthermore, on assuming a high-speed network, the propagation delay can be neglected.

Hassanein and Al-Otaibi [HaA00] proposed a routing algorithm that combines routing, CAC, and resource reservations. The proposed setup process employs a destination-controlled routing algorithm that uses the CAC criteria as its cost function. The proposed scheme is denoted the FRBS scheme, which has two stages, a Forward-Routing stage and a Backward-Setup stage. During the forward stage, routing information on all paths from source to destination is forwarded through setup messages to the destination using a controlled flooding approach. Setup messages include CAC information on all links within the traversed path. The destination node collects information on all possible paths in its routing table. The destination then makes a selection of the path with the best-cost value, and reservation is attempted along this path backwards towards the source such that the CAC criteria are met at this backward setup stage. The simulation results showed good performance in terms of accuracy of routing decisions and call blocking probabilities. However, the scheme is not scalable to large networks with many feasible paths.

A two-level forwarding mechanism based on the QoS requirements of the unicast flow is proposed by Ghosh et al. [GSA01]. All the routers in a network are assumed to be QoS aware, i.e., packets are forwarded based on both their destination and QoS requirements. Each connection request contains the destination ID, and the set of QoS requirements for that flow. The routing algorithm reads the destination and the QoS requirements, and returns a path (if available) that is most likely to satisfy the allocation requirements. The approach of Ghosh et al. [GSA01] is different than the approach of others [ChN98a, ChN98b, ChN98c, ShC00, and Hou96] in the sense that additional state information is used to reduce the overhead in connection establishment. A router stores information only about its immediate neighbors (routers reachable in one hop) and second-degree neighbors (neighbors of a neighbor). The advantage of this approach is two-fold. First, the message overhead and the impreciseness will not be as large as maintaining the global state. A router exchanges information only with its direct neighbors. As a result, the impreciseness in storing the information about the second-degree neighbors will not be as big as the impreciseness in storing the entire global state. Second, using the information about the second-degree neighbors, a router can forward the connection requests intelligently instead of blindly flooding the requests. This is because every router can now see two levels downstream. Hence, the overhead in connection establishment is reduced, even though additional overhead is incurred in maintaining the information about the second level links.

2.5 Summary

This chapter presented a review of various aspects of quality of service routing and related work. Routing algorithms used in *current Internet routing* protocols are mainly based on shortest path algorithms [RFC1583]. Shortest path algorithm can only find a least delay path or find a one with the most available resources. The goal of QoS routing algorithms is to find a path in the network that satisfies the given requirements of the application, and may also optimize the global network resource utilization [GSA01]. Depending on the scope of the path selection process, an algorithm either returns the next best hop (known as distributed routing) or the entire path to the destination (known as source routing). A brief description is presented on QoS routing strategies such as source routing and distributed routing strategies.

The Chapter 3 will present a detailed overview of the three routing algorithms we have selected as part of the forward routing stage for our proposed Integrated Routing Protocol. The three algorithms discussed in Chapter 3 are: (1) Delay-Constraint Unicast Routing (DCUR) algorithm, (2) Delay-Constraint Routing (DCR) algorithm, (3) Distributed Delay-Constraint Algorithm (DDCA).

The approach of Integrated Routing Protocol (IRP) is presented in Chapter 4. Similar to FRBS [HaA00], IRP is a distributed QoS-aware unicast delay-constraint routing protocol that works in two stages: a forward routing stage and a backward setup stage. The forward routing stage utilizes the above-mentioned three routing algorithms to find the QoS paths and resources are reserved along those paths in the backward setup stage if they follow the call admission criteria. The performance evaluation of IRP using a comprehensive simulation model is presented in Chapter 5. Simulation experiments are conducted in order to study the impact of call characteristics, such as network load, bandwidth requirement, delay bound and call setup time limit on the performance of IRP.

Chapter 3

Problem Formulation and Network Model

This chapter aims to formulate the problem of routing and identify routing information needed at each network node to achieve QoS over the Internet, in point-to-point communication channel for real-time traffic subject to an end-to-end delay constraint in connectionless IP networks. In literature, the unicast routing problem has been studied extensively and formulated as a Delay-Constrained Least-Cost (DCLC) path problem. This problem has been shown to be NP-complete [SRV97]. Several heuristics solutions can be found in literature for the DCLC problem [Wid94, SRV97, SuL98, ChN98c, ZKM01, BBB01].

The rest of this chapter is organized as follows. Section 3.1 describes how flows are handled in connectionless IP networks and how resources may be reserved for the duration of the flow. In Section 3.2 the problem description, network model design and routing information needed for QoS routing algorithms are presented. Section 3.3 describes our selected routing schemes by giving brief description and detail working mechanism of each scheme. Section 3.4 concludes this chapter.

3.1 Establishing Connection in IP Networks

Applications with stringent demands for QoS may require connection-oriented service. The need to reserve resources such as buffer space and bandwidth for each individual connection in order to guarantee its QoS requirement has made the connection establishment in high speed networks quite similar to call setup in circuit-switched networks [Hwa93]. In this thesis we emphasize such applications.

When a call (flow) request arrives with its specified QoS requirement, the routing algorithm chooses a path for the call from a set of possible paths and then proceeds as follows. First, the source node invokes the Call Admission Control (CAC) function to check if the new call can be admitted on the first outgoing link in the path. The call is accepted on the link if sufficient resources (residual bandwidth) are available on the link to meet the new call's QoS demand, while maintaining the agreed-upon QoS for the existing calls. If the call is accepted on the link, a certain amount of the bandwidth, determined by the resource reservation (RR) procedure will be allocated on that link for the duration of that call. The source node then passes the call request to its downstream neighbor on the chosen path. This neighbor then passes the request to its downstream neighbor if the QoS can be guaranteed at the next link also. This process continues until either the request is successfully passed to the destination node or the request fails to satisfy the QoS demand. In case of failure, the bandwidth that has been allocated for this call must be released and another path if available will be tried, if the call setup time has not yet expired. The call setup phase provides the guaranteed QoS required by the call.

3.2 Problem Description and Network Model

The algorithm that optimizes certain functions while providing a guaranteed upper bound on end-to-end delay in point-to-point communication channel are known as Delay-Constrained Least-Cost (DCLC) unicast routing algorithm. The Delay-Constrained Least-Cost (DCLC) problem finds shortest least-cost path from source node *s* to destination node *d*, such that the delay along the path does not exceed a delay constraint Δ bound. It is a constraint minimization problem that can be formulated as follows:

3.2.1 Delay-Constrained Least-Cost (DCLC) Problem

The network is modeled as a directed[2] graph $G = (V, E)$, where V is the set of nodes and E is the set of links. Each link $e = (i, j) \in E$ is associated with nonnegative cost value C(e) and a delay value D(e). The cost of a link can be assigned in various ways (e.g., link utilization, available bandwidth, etc.). Given a delay constraint Δ the problem is to find a path P from a source node ***s*** to a destination node ***d***, i.e., $P(s, d)$ such that to minimize the cost of the entire path without violating the delay on the upper bound. Mathematically, the DCLC problem is formulated as:

$$D(P) = \sum_{e \in P} D(e) \leq \Delta$$

Such that:

$$C(P) = \sum_{e \in P} C(e)$$

Cost is minimized over all paths satisfying the first condition.

At time the connection is requested, a minimum cost (shortest) path connecting a pair of end points is selected. Of course only paths consisting of edges with sufficient unused bandwidth may be chosen [Wax88].

3.2.2 Routing Information

Delay-constrained routing protocols require the following information be present at each node in the network. Each node v in the network maintains a delay table and a cost table

[2] Un-directed graph could also be used.

consisting of $|V - 1|$ entries (one entry for every other node). The entry for node **u** at node ***v*** consists of:

- The address of node u;
- The delay of the least-delay path from v to u, i.e., $D[P_{ld}(v,u)]$;
- The cost of the above path, i.e., $C[P_{ld}(v,u)]$; and
- The next hop of u on the LD path from v to u,i.e., ld_nhop$[P_{ld}(v,u)]$.

Similar information is maintained in the cost table but with the LD path replaced by the LC path.

- The address of node u;
- The delay of the least-cost path from v to u, i.e., $D[P_{lc}(v,u)]$;
- The cost of the above path, i.e., $C[P_{lc}(v,u)]$; and
- The next hop of u on the LC path from v to u, i.e., lc_nhop$[P_{lc}(v,u)]$.

The information in the delay and cost tables can be distributed to nodes using distance-vector (or link-state) protocols. In addition to cost and delay vectors, each node ***v*** maintains a routing table and a table that holds invalid routing information such as loop causing links and nodes. Every entry in the routing table corresponds to an established path from a source ***s*** to destination ***d*** that passes through node ***v***. Routing table entries are created during the connection establishment phase for a session involving real-time traffic that flows from source ***s*** to destination ***d***. In addition to the routing information, the routing table also holds information about the resources reserved for the connection from ***s*** to ***d***. When a real-time session terminates, the corresponding path is torn down, and all the routing entries corresponding to that session are deleted. An assumption is made that the cost vector and delay vector at all nodes are up-to-date, besides assuming that the link cost, the link delays, the contents of cost vector, the contents of delay vector do not change during the execution of routing algorithm [SRV97].

3.3 Selected Routing Schemes

This section describes the three routing schemes (DCUR, DCR, DDCA) we have chosen for comparison with our proposed routing scheme. All these schemes are delay-constrained unicast routing algorithms with one common objective, i.e., to minimize the path cost under the delay constraint.

3.3.1 Description of DCUR Scheme

Salama et al. [SRV97] proposed the Delay-Constrained Unicast Routing (DCUR) algorithm that was revised by Reeves and Salama [ReS00]. DCUR is a source-initiated algorithm that constructs a delay-constrained path connecting source node ***s*** to destination node ***d***. The path is constructed one node at a time, from source to destination. The DCUR scheme checks, if there exists a least cost path that keeps the delay under the constraint value. Then it is given two choices: either to select the next hop in LC path direction (denoted as lc_nhop) or select a next hop in LD path direction (denoted as ld_nhop). The source node (active node) after deciding the path direction sends request to the next hop node in that direction. Every node initially attempts to forward the packet to the next hop on LC path to the destination. However, if the least delay in the next hop (lc_nhop) is such that it is violating the delay constraint then the node attempts to forward the packet to the next hop along the LD path towards the destination.

DCUR requires only a limited amount of computation. The algorithm, by restricting the choice to only two nodes, fails to consider links that could potentially offer a better overall cost-delay performance. In addition, because of its reliance on cost and distance vector tables, the algorithm is dependent on the accuracy of these tables. For dynamic networks whose link parameters vary frequently, this accuracy cannot be guaranteed.

Working Mechanism of DCUR Scheme

DCUR assumes that no routing loops can occur. Figure 3.1 shows the algorithm for source node. At line 1, the source node initiates the path construction between a source and destination node with delay constraint Δ. Lines (2–3) states, if the destination node is already reached or the least delay value from source to destination violates Δ, then stop routing execution and send a failure message as there exists no delay constraint path. Otherwise, continue the forward routing process and initialize the required parameters (lines 4- 8). The algorithm forces all the nodes to take the LC path if the delay constraint is satisfied otherwise the LD path is used. At line 9, if the path_direction taken is LC then it sets the next_node to lc_nhop from active_node towards d. If the path_direction is LD, then next_node is set to ld_nhop from active_node towards d (lines 11- 12). If the next hop node is the same on both LC path and the LD path from active_node to the destination, then path_direction (by default) will be set to LD (lines 13-14). In line 15, a Path_Construction message is send along the LC path. A Call_Setup_Req message for a connection request is sent for this pair of ***s*** and ***d***. Details of the Call_Setup_Req are given in Chapter 4.

```
1.  Inititate_Path_Construct (s, d, $\Delta$) // DCURScheme
2.  if (s == d) or (D[$P_{ld}$(s, d)] $\geq \Delta$)
3.      stop execution and send failure message
4.  else   //continue forward routing stage
5.      active_node = s
6.      prev_node = NULL
7.      delay_so_far = 0
8.      cost_so_far = 0
9.      if path_direction = LC
10.         next_node = lc_nhop(active_node, d)
11.     if path_direction = LD
12.         next_node = ld_nhop(active_node, d)
13.     if (lc_nhop = ld_nhop)
14.         path_direction = LD  //LD by default
15.         send Path_Construction(s, d, prev_node, delay_so_far, cost_so_far, $\Delta$)
        // send Call_Setup_Req for s and d
16. end if
```

Figure 3.1 DCUR algorithm for source node

Figure 3.2 shows the DCUR algorithm for intermediate node. Lines 1 to 6 state that upon receiving the Path_Construction message, a node checks if the destination node has been reached, the routing table entry is created and DCUR_Path must be return with computed path_ delay and path_cost. An ACK message is send to reserve resources for this pair of source and destination nodes in order to establish a connection (as is shown in line 3). The details of ACK message are presented in Chapter 4. Otherwise, the algorithm will continue routing in the LC direction. The algorithm constructs the path hop-by-hop. At each hop, a delay is computed from the active_node to the next node along least cost (lc_nhop) path (line 13 and 14) and tested as follows:

If (delay_so_far + path_delay + D[P_{lc}(lc_nhop, d)] $\leq \Delta$)

Where path_delay is:

path_delay = D[P_{lc}(active_node, lc_nhop)] //along lc_nhop

Similarly, for the same node, cost is computed along lc_nhop (line 15). If the path satisfies the delay-constraint along lc_nhop, then active_node creates routing table entry and sends Path_Construction message to next_node along LC (line 20). Otherwise, path_direction is switched to LD (line 22). If least delay (ld_nhop) path direction is taken (lines 24-32), delay and cost are computed in similar fashion as along the lc_nhop, except that this time it is for ld_nhop. Routing continues along ld_nhop direction, creating routing table entry and sending Path_Construction messages, until destination node is reached.

The path constructed by existing distance-vector protocols are guaranteed to be loop free if the contents of the distance vectors at all nodes are up-to-date and the network is in stable condition. However, up-to-date cost vector and delay vector contents and stable network conditions are not sufficient to guarantee loop-free operation for DCUR. In DCUR, each node involved in the path construction operation selects either the LC path direction or the LD path direction, or all nodes choose the LD path direction, then no loops can occur, because the resulting paths are the LC path or LD path respectively. However, if some paths choose the LC path direction while others choose the LD path direction, loops may occur. Loops can be removed using loop-elimination mechanism. The message complexity of DCUR is $O(V^2)$, where V is the number of nodes in the network.

```
1.  Path_Construction(s, d, prev_node, delay_so_far, cost_so_far, $\Delta$)
2.  if (active_node == d) // we've reached destination already
3.      //create routing table entry and d enters backward setup process by sending ACK to s
4.      path_delay = delay_so_far
5.      path_cost = cost_so_far
6.      return DCUR_Path(s, d, active_node, path_delay, path_cost, $\Delta$, path_direction)
7.  else  //continue forward routing process
8.      if (lc_nhop = ld_nhop)
9.          path_direction = LD  // path direction by default is set to LD
10.     else
11.         path_direction = LC    //path direction is force to set at LC
12. end if
13. if path_direction = LC  //then compute delay and cost along LC path
14.     path_delay = D[$P_{lc}$(active_node, lc_nhop)]  //along lc_nhop
15.     path_cost = C [$P_{lc}$(active_node, lc_nhop)]   //along lc_nhop
16.     delay_so_far = delay_so_far + path_delay + D[$P_{lc}$(lc_nhop, d)] $\leq \Delta$
17.     if (delay_so_far + path_delay + D[$P_{lc}$(lc_nhop, d)] $\leq \Delta$)
18.         cost_so_far = cost_so_far + path_cost + C[$P_{lc}$(lc_nhop,d)]
19.         //create routing table entry and continue routing along lc_nhop
20.         send Path_Construction(s,d, prev_node, delay_so_far, cost_so_far ,$\Delta$)
            // send Call_Setup_Req for s and d
21.     else
22.         path_direction = LD  // switch to ld_nhop if lc_nhop no more satisfies $\Delta$
23.     end if
24. if path_direction = LD  //then compute delay and cost along LD path
25.     path_delay = D[$P_{ld}$(active_node, ld_nhop)]  //along ld_nhop
26.     path_cost = C[$P_{ld}$(active_node, ld_nhop)]   //along ld_nhop
27.     delay_so_far = delay_so_far + path_delay + D[$P_{ld}$(ld_nhop, d)]$\leq \Delta$
28.     if (delay_so_far + path_delay + D[$P_{ld}$(ld_nhop, d)]$\leq \Delta$)
29.         cost_so_far = cost_so_far + path_cost + C[$P_{ld}$(ld_nhop, d)]
```

```
30.  //create routing table entry and continue routing along ld_nhop
31.      send Path_Construction(s,d, prev_node, delay_so_far, cost_so_far ,$\Delta$)
         // send Call_Setup_Req for $s$ and $d$
32. end if
```

Figure 3.2 DCUR algorithm for intermediate node

3.3.2 Description of DCR Scheme

Sun and Langendorfer [SuL98] proposed distributed Delay-Constrained Routing (DCR) algorithm, which is similar to DCUR with some differences. When DCR finds and selects the LC path, it gives that information to the next node so that the next node does not check again for LC path. The DCR may generate the same path as DCUR but it is less complicated and efficient in terms of path selection algorithm than DCUR, because, it does not check for the LC path every time.

Working mechanism of DCR Algorithm

The DCR scheme also finds routes in hop-by-hop fashion like the DCUR scheme. Figure 3.3 shows the algorithm for source node. At line 1, the source node initiates the path construction for a pair of source and destination nodes with required delay constraint as Δ. Lines 2–3 state that if the destination node is already reached or the least delay value from source to destination dose not follow the Δ, i.e., if $D[P_{ld}(s,d)] > \Delta$, then stop routing execution and send a failure message, as there exist no delay constraint path. Otherwise, continue the forward routing process and initialize the required parameters (lines 5- 8). The source node further checks (line 9) from its cost vector if delay along least cost path from s to d satisfies the delay constraint, i.e., if $D[P_{lc}(s,d)] < \Delta$ then sets (line10) next_node to lc_nhop from source to the destination. The path_direction is set to LC (line 11). A Path_Construction message is send along the LC path (line 12).

```
1.  Initiate_Path_Construct(s, d, $\Delta$) //DCRScheme
2.  if (s == d) or (D[$P_{ld}$(s,d)] > $\Delta$)
3.     stop execution and send failure message
4.  else    // continue forward routing stage
5.      active_node = s
6.     prev_node = NULL
7.     delay_so_far = 0
8.     cost_so_far = 0
9.     if D[$P_{lc}$(s,d)] < $\Delta$
10.        next_node = lc_nhop[$P_{lc}$(s, d)]
11.        path_direction = LC
12.        send Path_Construction(s,d, prev_node, delay_so_far, cost_so_far ,$\Delta$)
           // send Call_Setup_Req for s and d
13.    else
14.        next_node = ld_nhop[$P_{ld}$(s, d)]
15.        path-direction = LD
16.        delay_so_far = D(s, ld_nhop[$P_{ld}$(s, d)])
17.        cost_so_far = C(s, ld_nhop[$P_{ld}$(s, d)])
18.        send Path_Construction(s,d, prev_node, delay_so_far, cost_so_far ,$\Delta$)

19.         // send Call_Setup_Req for s and d
20.     end if
20.    send Path_Construction(s,d, prev_node, delay_so_far, cost_so_far ,$\Delta$)
       // send Call_Setup_Req for s and d
21. end if
```

Figure 3.3 DCR algorithm for source node

A connection request (i.e., Call_Setup_Req) for this pair of *s* and *d* will establish a real time channel. Otherwise (lines 13-17), *s* reads next hop node on the least delay path from its delay vector and sets next_node as ld_nhop from source to destination node. A path_direction is set to LD. A Path_Construction message is send along the LD path (line 18). A Call_Setup_Req message for a connection request is sent for this pair of ***s*** and ***d***. Details of the Call_Setup_Req are given in Chapter 4.

Figure 3.4 shows an algorithm for intermediate nodes in the DCR scheme. Lines 1-4 state that upon receiving the Path_Construction message, if destination node has been reached, a routing table entry is created and DCR_Path must be returned. An ACK message is send to reserve resources for this pair of source and destination node in order to establish a connection (as is shown in line 3). The details of ACK message are presented in Chapter 4. Otherwise, the active_node continues forward routing and constructs DCR path hop-by-hop.

If the path_direction is LC or if delay_so_far + D[P_{lc}(active_node ,d)] < Δ then the active_node reads the next hop node on the least cost path (lc_nhop) towards ***d*** from its cost vector and sets the next_node to lc_nhop[P_{lc}(active_node, d)] and path_direction = LC (lines 6-9). The cost_so_far is computed as well for the same path. If the path satisfies the delay-constraint along lc_nhop, then active_node creates a routing table entry and sends Path_Construction message to next_node along LC (line 10-11). A Call_Setup_Req message for a connection request is sent for this pair of ***s*** and ***d***. However, if the path_direction is LD in the received Path_Construction message instead of LC, then the active_node reads the next hop node on the least delay path (ld_nhop) towards ***d*** from its delay vector and sets the next-node to ld_nhop[P_{ld}(active_node, d)] and path-direction = LD (lines 13-16).

```
1.  Path_Construction(s, d, prev_node, delay_so_far, cost_so_far, $\Delta$)
2.  if (active_node == d) // we've reached destination already
3.    //create routing table entry and d enters backward setup process by sending ACK to s
4.     return DCR_Path(s, d, active_node, delay_so_far, cost_so_far, $\Delta$, path_direction)
5.  else   // continue routing process
6.     if (path_direction = LC) or (delay_so_far +  D[$P_{lc}$(active_node ,d)] < $\Delta$ )
7.        next_node = lc_nhop[$P_{lc}$(active_node, d)]
8.        cost_so_far = cost_so_far + C[$P_{lc}$(active_node ,d)]
9.        path_direction = LC
10.       // create routing table entry and continue routing along lc_nhop
11.       send Path_Construction(s, d, prev_node, delay_so_far, cost_so_far, $\Delta$)
          // send Call_Setup_Req for s and d
12.    else
13.       next_node  = ld_nhop[$P_{ld}$(active_node, d)]
14.       cost_so_far = cost_so_far + C[$P_{ld}$(active_node ,d)]
15.       delay_so_far = delay_so_far + D[active_node, ld_nhop(active_node,d)]
16.       path_direction = LD
17.       // create routing table entry and continue routing along ld_nhop
18.       send Path_Construction(s, d, prev_node, delay_so_far, cost_so_far, $\Delta$)
          // send Call_Setup_Req for s and d
19.    end if
20.    // create routing table entry and continue routing along ld_nhop
21.    send Path_Construction(s, d, prev_node, delay_so_far, cost_so_far, $\Delta$)
      // send Call_Setup_Req for s and d
22. end if
```

Figure 3.4 DCR algorithm for intermediate node

The cost_so_far is computed as well for the same path. If the path satisfies the delay-constraint along ld_nhop, then active_node creates routing table entry and sends Path_Construction message to next_node along LD (line 18). Routing continues along ld_nhop direction, creating routing table entry and sending Path_Construction messages, until the destination node is finally reached.

If the destination ***d*** receives path-construction message it means the delay-constrained path has been successfully constructed. The destination node ***d*** sends an ACK message back to source node ***s*** to reserve resources (RR) on the way back towards the source node. When source node receives ACK message, it marks the end of the routing and call setup process. A connection is setup on a least-cost least-delay path from ***s*** to ***d*** with desired QoS resources.

3.3.3 Description of DDCA Scheme

Zhang et al. [ZKM01] proposed a distributed heuristic called distributed delay-constrained algorithm (DDCA). DDCA is based on the relay strategy[3]. In essence, DDCA is an extension of the DCR algorithm by [SuL98], which uses a similar procedure to construct a delay-constrained path from a source node to a destination node. The reservation message travels along the LD path until reaching a node from which the delay of its LC path satisfies the delay constraint. From that node and on, the message travels along the LC path all the way to the destination. In DDCA, each node in the network maintains a delay table and a cost table. Similar tables are maintained in algorithms

[3] The node that connects two super edges on the selected path is called the relay node, where the super edge is defined as a connected segment of the path on which all routers use the same routing metric (either delay or cost) for packet forwarding.

DCUR and DCR, each consisting of $|V - 1|$ entries (one entry for every other node). The routing information maintained at each node are described in Section 3.2.3

Working mechanism of DDCA Algorithm

The DDCA algorithm initially checks the feasibility of the LC path from ***s*** to ***d***. At line 1 in Figure 3.5, the source node initiates the path construction for a pair of source and destination nodes with total_cost and required delay constraint as Δ. Lines 2–3 state that if the destination node is already reached or the least delay value from source to destination does not follow the Δ, i.e., if $D[P_{lc}(s,d)] > \Delta$, then stop routing execution and send a failure message, as there exist no delay constraint path. If $D(P_{lc}(s,d)) \leq \Delta$, the algorithm returns this path $[P_{lc}(s,d)]$. Otherwise, the algorithm checks if a feasible path is available (by verifying that $D[P_{ld}(s,d)] \leq \Delta$) and continue forward routing process and initialize the required parameters (lines 6- 9).

The source node sets next_node to ld_nhop from source to the destination (line10). The path_direction is set to LD (line 11). Initial values computed for delay_so_far and cost_so_far along ld_nhop are shown in lines 12 and 13. A Path_Construction message is send along the LD path (line 14). Otherwise (lines 15-19), ***s*** reads next hop node on the least delay path from its cost vector and sets next_node as lc_nhop from source to destination node. The path_direction is set to LC. A Path_Construction message is send along the LC path (line 20). Similar to DCUR and DCR schemes, Path_Construction messages are generated and send from source **s** to **d** along LD and LC path_direction and Call_Setup_Req messages are send for pair of ***s*** and ***d*** to establish real-time connections with the required QoS parameters. Details of connection establishment can be found in Chapter 4.

```
1.  Initiate_Path_Construct(s, d, total_cost, $\Delta$)  //DDCA Scheme
2.  if (s == d) or (D[$P_{lc}$(s,d)]  > $\Delta$)
3.      stop execution and send failure message
4.  else    // continue forward routing stage
5.      if  D[$P_{ld}$(s,d)] $\le \Delta$
6.          active_node = s
7.          relay_node = NULL
8.          delay_so_far = 0
9.          cost_so_far = C[$P_{ld}$(s, d)]
10.         next_node = ld_nhop[$P_{ld}$(s, d)]
11.         path-direction = LD
12.         delay_so_far = D(s, ld_nhop[$P_{ld}$(s, d)])
13.         cost_so_far = C(s, ld_nhop[$P_{ld}$(s, d)])
14.         send Path_Construction(s,d, relay_node, delay_so_far, cost_so_far, total_cost, $\Delta$)
            // send Call_Setup_Req for s and d
15.     else
            next_node = lc_nhop[$P_{lc}$(s, d)]
16.         path_direction = LC
17.         delay_so_far =  D(s, lc_nhop[$P_{lc}$(s, d)])
18.         cost_so_far = C(s, lc_nhop[$P_{lc}$(s, d)])
19.         send Path_Construction(s,d, relay_node, delay_so_far, cost_so_far, total_cost, $\Delta$)
            // send Call_Setup_Req for s and d
20.     end if
21.     send Path_Construction(s,d, relay_node, delay_so_far, cost_so_far, total_cost, $\Delta$)
        // send Call_Setup_Req for s and d
22. end if
```

Figure 3.5 DDCA algorithm for source node

Figure 3.6 shows the algorithm at intermediate nodes in the DDCA scheme. Lines 1-4 state that upon receiving the Path_Construction message, if destination node has reached, a routing table entry is created and DDCA_Path must be returned. An ACK message is send to reserve resources for this pair of source and destination node in order to establishment a connection (line 3). Otherwise, the active_node continues forward routing and construct DDCA path hop-by-hop.

If path_direction taken is LD and if delay_so_far + $D[P_{lc}(\text{active\_node}, d)] \leq \Delta$, then the active_node has found a relay_node. A relay_node is a node with better path cost than LD path, i.e., if cost_so_far + $C[P_{lc}(\text{active\_node}, d)]$ < total_cost, therefore the next_node is set to a relay_node (line 9). The path cost is computed as follows and this least cost path will be returned: total_cost = cost_so_far + $C[P_{lc}(\text{active\_node}, d)]$. Path_Construction message will be send along this path. However, if the active_node is not a relay_node (lines 14-16), delay_so_far and cost_so_far will be computed along the LD path as follows:

delay_so_far = delay_so_far + D(active_node, ld_nhop)

cost_so_far = cost_so_far + C(active_node, ld_nhop)

A Path_Construction message is send in search of a better path in the LD path direction. If the path_direction is LC (line 19) and if delay_so_far + $D[P_{ld}(\text{active\_node}, d)] \leq \Delta$, then the active_node has found a relay_node. A relay_node is a node with better path cost than LC path, i.e., if cost_so_far + $C[P_{ld}(\text{active\_node}, d)]$ < total_cost, therefore the next_node is set to a relay_node (line 22). The path cost is computed as follows and this least cost path will be returned: total_cost = cost_so_far + $C[P_{ld}(\text{active\_node}, d)]$. If LD path violates delay constraint (line 20) then delay_so_far and cost_so_far are computed along LC path as follows (lines 25-26):

delay_so_far = delay_so_far + D(active_node, lc_nhop)

cost_so_far = cost_so_far + C(active_node, lc_nhop)]

```
1.  Path_Construction(s, d, relay_node, delay_so_far, cost_so_far, total_cost, $\Delta$)
2.  if (active_node == d ) // we've reached destination already
3.      //create routing table entry and d enters backward setup process by sending ACK to s
4.      return DDCA_Path(s, d, active_node, delay_so_far, cost_so_far, total_cost, $\Delta$,
    path_direction)
5.  else  // continue forward routing process
6.      if path_direction = LD
7.          if delay_so_far + D[$P_{lc}$(active_node, d)] $\leq \Delta$ //active_node is a relay_node
8.            if cost_so_far + C[$P_{lc}$(active_node,d)] < total_cost //path is better than LD path
9.                relay_node = active_node
10.               total_cost = cost_so_far + C[$P_{lc}$(active_node,d)]
11.           end if
12.           //create routing table entry, return DDCA_Path and continue routing along
    ld_nhop
13.           send Path_Construction(s,d,active_node,delay_so_far,cost_so_far,total_cost,$\Delta$)
             // send Call_Setup_Req for s and d
14.         else  //active_node is not a relay_node
15.             delay_so_far = delay_so_far + D(active_node, ld_nhop)
16.             cost_so_far = cost_so_far + C(active_node, ld_nhop)
17.             send Path_Construction(s,d,relay_node,delay_so_far,cost_so_far, total_cost, $\Delta$)
                // send Call_Setup_Req for s and d
                // continue forward routing along ld_nhop
18.         end if
19.     else  //path_direction = LC
20.         if delay_so_far + D[$P_{ld}$(active_node, d)] $\leq \Delta$ //active_node is a relay_node
21.             if cost_so_far + C[$P_{ld}$(active_node,d)] < total_cost //better path is found
22.                 relay_node = active_node
23.                 total_cost = cost_so_far + C[$P_{ld}$(active_node,d)]
24.             end if
25.             delay_so_far = delay_so_far + D(active_node, ld_nhop)
26.             cost_so_far = cost_so_far + C(active_node, ld_nhop)]
27.             //create routing table entry, return DDCA_Path and continue routing along
    lc_nhop
28.             send Path_Construction(s,d,active_node,delay_so_far,cost_so_far, total_cost,$\Delta$)
                // send Call_Setup_Req for s and d
29.         else  // active_node is not a relay_node, terminate unsuccessfully
30.             return DDCA_Path(s, d, active_node, delay_so_far, cost_so_far, total_cost, $\Delta$,
    path_direction)
31.         end if
32. end if
```

Figure 3.6 DDCA algorithm for intermediate node

A Path_Construction message is sent in search of a better path in LC path direction. A search for a better path continues in LC direction, however, it terminates unsuccessfully (line 29-32).

Routing continues along ld_nhop (lc_nhop) direction, creating routing table entry and sending Path_Construction messages, until the destination node is finally reached. This marks the end of forward routing stage. If the destination ***d*** receives Path-Construction message it means the delay-constrained path has been successfully constructed. The destination node ***d*** sends an ACK message back to source node ***s*** to reserve resources on the way back towards the source node. When the source node receives ACK message, it marks the end of the routing and call setup process. A connection is set up on a least-cost least-delay path from ***s*** to ***d*** with desired QoS resources. The worst-case message complexity of DDCA is O(|V|).

3.4 Summary

The main goal of this thesis is to test the viability of destination-controlled Delay-Constraint Least-Cost (DCLC) QoS routing. For this purpose, we investigated three proposed heuristics solutions for DCLC problem, namely DCUR, DCR and DDCA. All the three heuristics work more or less in the same fashion, with the common objective of cost minimization under certain delay constraint.

The DCUR scheme checks if there exists a least cost path that keeps the delay under the constraint value. If there is such a path that satisfies the delay constraint in the least-cost path direction, it selects the next hop in that direction otherwise the least delay path's next hop is selected. The routing entry of the path is saved in the routing table of the active node and the path construction responsibility is delegated to the next hop. Since once the least cost path node is found which satisfies the delay constraint there will always be a least cost node along that path that satisfies the same condition. DCUR hence selects an LC path at each hop until it finds the LD path node satisfying the delay constraint condition. Afterwards, it selects the LD path.

The DCR scheme is similar to DCUR with a slight difference. When it finds and selects the LC path it gives that information to the next node so that the next node does not check again for the LC path. The path generated by DCR might be the same as DCUR but it is less computationally expensive in terms of the path selection algorithm than DCUR.

The DDCA scheme makes use of the fact that by selecting the LC path nodes first (until the LC path can no longer satisfy the delay constraint) and LD path later. This results in the path that has less cost than the path generated by DCUR or DCR while still satisfying the delay constraint. Otherwise, it selects the same path as the previous algorithms. The routing algorithms should be able to efficiently manage the network resources such as residual bandwidth, and buffer space. The cost of a link is taken as a function of link's bandwidth utilization. Cost performance should be the criteria for algorithms in efficiently managing the residual link bandwidth.

Chapter 4

Integrated Routing Protocol Overview

Multimedia applications in high-speed networks require acceptable performance independent of traffic conditions. To provide performance guarantees in delivering such applications, resource reservation becomes indispensable. A good routing scheme must be used to find a feasible path before any allocation of network resources can take place.

The routing schemes considered in this research (DCUR, DCR, DDCA) are mainly focusing on the routing aspect of the problem while leaving the CAC and resource reservation problems for future investigation. This thesis analyzes, compares and interweaves the above-mentioned distributed QoS routing schemes with call admission and resource reservation in order to find the most viable path finding algorithm among the chosen schemes. We call this approach the *Integrated Routing Protocol (IRP).* IRP operates in two stages; a forward routing stage and a backward setup stage. During the forward routing stage, the routing information is forwarded from source towards destination on feasible paths through call setup messages. A successful path generation through each of the chosen candidate routing schemes in parallel marks the completion of forward routing stage. The destination node collects information on feasible paths through chosen distributed routing algorithms in its database. The backward call setup stage starts after the destination node chooses a viable path for a single source and destination node pair. The call setup messages include the latest CAC information on links within the traversed paths. A viable least-cost path with the most available bandwidth within the delay constraint is then selected. Connection setup attempts resource allocation backwards towards the source node, such that the CAC criteria are met. The performance measures studied through the simulation model are the call acceptance ratio (CAR) and call blocking ratio due to lack of bandwidth (CBRLB) and call blocking ratio due to delay bound violation. We also studied the effect of call setup time limit on the IRP scheme.

The rest of the chapter is structured as follows. The problem definition of Integrated Routing Protocol (IRP) is presented in Sections 4.1. Section 4.2 presents a detailed overview of the IRP scheme through defining the messages exchanged between various nodes in order to establish a real time connection. A pseudo-code description of the algorithm integrated with call admission and resource reservation is presented. Summary of the chapter is presented in Section 4.3.

4.1 Integrated Routing Protocol Problem Definition

A directed network is modeled as a set of N nodes that are interconnected by a set of E communication links. Each node ***i*** has a routing table that has an entry for every other node ***j***. The distance-vector algorithm maintains the routing table. It is assumed that links have different cost and delay values in each direction. The end-to-end QoS provision is depicted as follows: when a source node ***s*** wants to send a data packet with a certain QoS requirement to a destination node ***d***, it issues a QoS connection setup request. The system first identifies a feasible path between sending and receiving ends that can satisfy the QoS requirement. Then the destination node checks the availability and reserves resources along the path to establish the desired connection. After the connection is established, the source node sends its data packets along that path to the destination node. The reserved resources along the path guarantee the quality of service. Resources should not be reserved during QoS routing because the routing may involve a large portion of network instead of a single path from the source to the destination [HaA00].

The IRP scheme works in two stages; the forward routing stage and a backward setup stage. The forwarding routing stage of the scheme assumes $P_1(s, d)$, $P_2(s, d)$ *and* $P_3(s, d)$ are three (one for each of our candidate routing algorithms – DCUR, DCR and DDCA) least-cost paths between single pair of source node ***s*** and a destination node ***d*** that are determined through call setup messages and meet the end-to-end delay requirements. The destination node will collect information about all the three paths (P_1, P_2, *and* P_3) determined by the respective schemes. Following a minimum cost criteria, the destination node chooses a path with lowest cost as the best viable path that obeys the delay bound constraint. The backward setup stage of the IRP scheme will be invoked by the time the destination node receives the path information. It will send a call setup message along the received path backward towards the source node. The call setup message tests whether any one of the chosen paths P_1, P_2 *or* P_3 is able to meet the CAC criteria and end-to-end delay bound requirements for the connection under consideration. On finding such path, the destination node efficiently allocates resources from itself backwards towards the source node along the selected path to establish a successful connection. The problem of Integrated Routing Protocol (IRP) can be formally stated as follows:

Given a connection and the path P between source node s and destination node d, based on the available resources, determine whether the connection can be admitted on the path P and if so, reserve the necessary resources, assuming that the desired QoS is determined by the destination.

If a connection can be successfully established on the chosen viable path within the specified call setup time limit, the control is transferred back to the destination node for the selection of next best viable path. Resource reservation is attempted along this path backwards towards the source node, within a connection setup time limit. In case of failure in connection establishment phase, control will be transferred back to the destination node with a failure message and will release the already allocated resources along the partially established path.

The selection of proper routing metric is an important factor in QoS routing. The IRP utilizes buffering and propagation delays as the delay metric and reserved bandwidth as the cost metric. The delay experienced in the network consists of propagation delays and buffering delays. The buffering delay on a link is assumed to be a dynamic metric that changes as the link bandwidth changes. The buffering delay reflects the reserved bandwidth and traffic characteristics on the network [AlO98], where higher network loads can cause data streams transferred on the network to experience longer delays due to increased buffering.

4.2 Integrated Routing Protocol Overview

The objective of integrated routing approach is to provide and evaluate alternate paths in unicast routing in order to enhance the chances of call acceptance, success ratio, network throughput and robustness. An overview of integrated unicast destination-control routing algorithms with CAC and RR using the FRBS approach is described in detail in this section.

4.2.1 Messages Exchanged by IRP

The following are the messages used in our proposed IRP scheme:

1. **Call_Setup_Req (Call_ID, Req_B, Δ, μ, delay_so_far, path_info):** a unicast setup message issued at the destination node towards the source node after routing protocol finds a feasible path in order to establish a real time connection with the following parameters:
    - **Call_ID**: a system wide unique identifier is assigned to every call and is maintained by a counter value, which increments by one every time a new call request is issued at the destination node. There can be many connection requests arriving at the same period of time. Messages belonging to different requests are sent over the network simultaneously. They are distinguished by their Call IDs, and no message interference will occur among different requests.
    - **QoS requirements of a Call:**
        a. **Req_B** is the connection bandwidth requirement, which is the bandwidth lower bound,
        b. Δ is the end-to-end delay bound and

c. μ is the setup time delay.

- **delay_so_far**: the delay of the traversed path from the source to the current node.
- **path_info**: the feasible path from source node to the destination

2. **ACK (Call_ID, Req_B, block_links, path_info):** a reservation message that contains the connection required bandwidth (Req_B) and the path_info from the routing table, which represents the path chosen to connect the destination node with the source node for a specific connection begin identified by its Call_ID. The ACK message will reserve the desired resources only if a node passes the CAC test. Each time the ACK message is sent out it will check a list of block_links so that to avoid loops and broken links. Where block_links is a table storing information about broken links.
3. **NACK (Call_ID, path_info, block_links):** a negative acknowledgment message when issued will release the resources along the path being stored at path_info. It also contains block_links, which are assumed to be a list of all blocked links encountered in earlier attempts to setup the connection. A NACK message is an indication of resource reservation failure and triggers the destination to select another feasible path. NACK releases the partially allocated resources before it triggers the destination to select a new viable path.

IRP utilizes the following Call Admission Control (CAC) criteria:

1. **(dly_so_far $\leq$ Δ)**: where dly_so_far is the accumulative delay along the path till the current node and is tested against the upper limit on the delay.

2. **(b(e)+ Req _B < CB):** where b(e) is the reserved bandwidth on a link and Req_B is the connection QoS requirement which is tested against the upper limit on bandwidth, i.e., Cost Bound (or CB).

If either of the CAC tests fails on a link, it will not be added to the path. Such failed links will be recorded in a block_links table. The block_links table will be checked each time an ACK message is sent out so as to avoid loops and broken links.

4.2.2 Operation of Integrated Routing Protocol

The Integrated Routing Protocol (IRP) utilizes the FRBS approach and works in two stages, the Forward Routing Stage and Backward Setup Stage.

4.2.2.1 Forward Routing Stage

The forward routing stage starts when an application sends out a routing request and proceeds as follows:

A pair of source and destination nodes will enter the routing process. The routing process will be carried out by all of our chosen routing algorithms in parallel for the same pair of source and destination nodes. All the three routing algorithms will choose a least-cost path that follows the delay bound constraint and store their respective paths (P1, P2 and P3) along with their respective costs (C1, C2 and C3) in a table at the destination node, as shown in Figure 4.1.

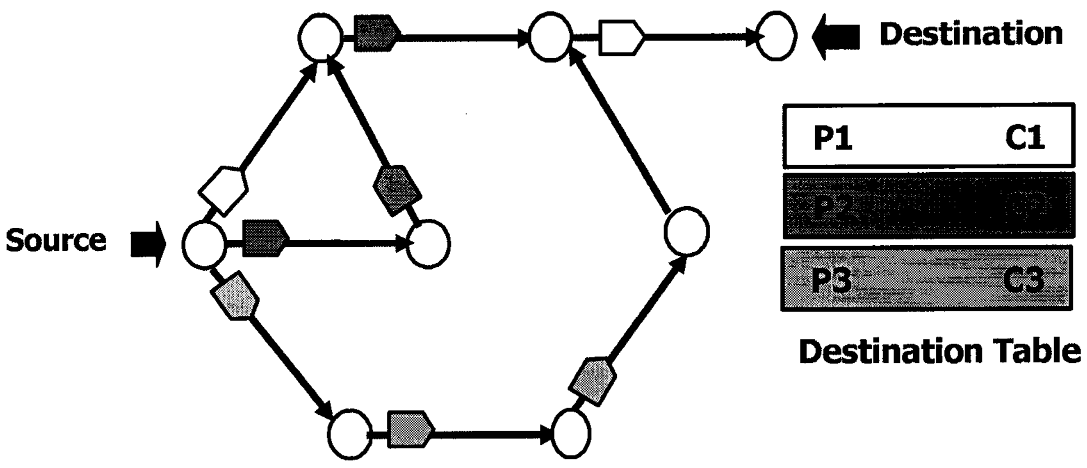


Figure 4.1 IRP at Forward Routing Stage

The forward routing process ends when all possible paths of all selected schemes are forwarded through call setup request messages towards the destination node, requesting a connection establishment. The Call_Setup_Req message carries the most up-to-date CAC information on the links within the traversed paths.

4.2.2.2 Backward Setup Stage

The destination node will enter a backward setup stage upon receiving the Call_Setup_Req from the source node. At the destination node the backward connection configuration process proceeds as follows:

The destination node in IRP will have feasible paths stored at its database being selected through all of the three distributed routing schemes. The most viable path selection decision is made at the receiver end by looking up at its database. A least cost path that obeys the end-to-end delay bound will be selected as the most viable path. The destination node may delay the selection of a path for a waiting time period τ. The philosophy behind the waiting time period is that there may be some other paths under generation process and coming towards the destination node to be stored at the routing table [KhH03]. We do not want to eliminate the possibility of such paths from consideration. The waiting period is calculated based on the threshold limit of the application and the call characteristics [AlO98]. An acknowledgement (ACK) message will be sent out on the most viable path from the destination backward towards the source node as shown in Figure 4.2, where P2 is selected as viable path. Desired resources are allocated along the path P2 within the setup time limit μ and delay bound constraint Δ.

If a call request is rejected or the connection times out when the setup time limit expires for a session while allocating resources backwards from destination towards source node, a negative acknowledgement (NACK) message will be issued from the failed node. The partially (or fully) constructed path will be torn down, and the resources allocated so far will be released and made available for other flows. The control will be transferred back to the destination from the node that encountered a failure, in order to select the next viable path being generated by another scheme from the database at the destination node. A new ACK message will be released for RR along this newly selected path. A connection will be established when ACK message reaches the source node. The ACK message is forwarded to every node in the reverse path in order to reserve requested resources. If any link violates CAC condition, when the resources are reserved by other connections, the algorithm control will go back to the destination node, after releasing reserved resources from the path fragment that has been reserved during backward process. The destination node then chooses another path, which does not contain previously encountered blocked links.

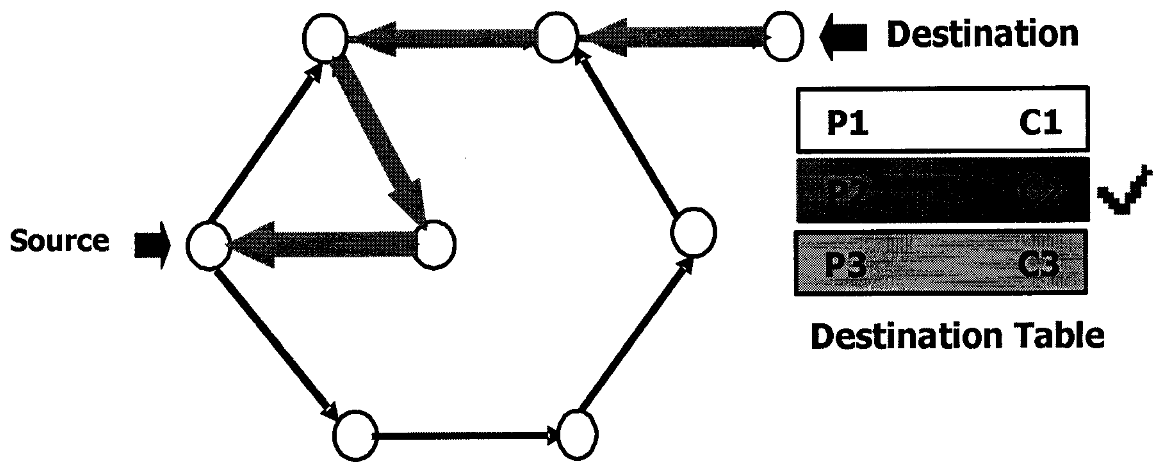


Figure 4.2 IRP at Backward Setup Stage

When the source node receives the ACK message, it knows that a reserved path has been established to the destination. However, the destination node does not learn of the connection establishment until it receives the first data packet. At this time the destination node removes all of the unnecessary information corresponding to this connection from its routing table.

The objective of routing prior to resource allocation is to select a path (that has been determined by our selected routing algorithms) with best chance of satisfying the resource requirement. After selecting such a path (viable path), resource reservation is attempted successfully along the path, thus eliminating the chances of selecting paths that would have otherwise been resulted in congested traffic.

4.2.3 Pseudo-Code Description of IRP

The following section presents a pseudo code description of our Integrated Routing Protocol. Various control messages are exchanged between nodes along a path in a distributed manner. The source node, intermediate nodes and destination node react differently to messages exchanged in the protocol. The three control messages of the IRP are Call_Setup_Req, ACK and NACK. Operation of each node is triggered by the arrival of a control message.

4.2.3.1 Algorithms for Source Node

The source node algorithm at the forward routing stage is shown in Figure 4.3a. Line 1 represents the beginning of the forward routing stage where path construction will begin by selecting a routing scheme. Lines 2-3 provide a test for the continuation of the forward routing process. If the source node already reaches a destination or the least-delay value from the source to the destination node exceeds the delay bound, then there exists no such path that satisfies the delay constraint therefore execution of routing must stop. Otherwise, the source node first checks if the call setup time (μ) is exhausted. In that case the setup message at the source node would discard the request (line10). The routing begins to forward Call_Setup_Req message on the selected outgoing links, if the call admission test is successful and the setup time is not expired (lines 5-6). The outgoing links are selected by the routing algorithms (DCUR, DCR, and DDCA). These algorithms are run in parallel to find the most feasible least-cost path. A connection establishment request to the next available node is then sent out (line 7). The Call_Setup_Req message carries the call QoS requirement, the delay and the maximum setup time limit a call can tolerate and a list of all the nodes on the shortest path to the destination.

Figure 4.3b shows the algorithm for source node during the backward setup stage. Lines 13-21 indicate that if the call setup time is not exhausted and there are enough resources

```
1.  // Initiate path construction using the available routing schemes i.e., DCUR, DCR, DDCR
2.  if (s == d )or (D[$P_{ld}$(s,d)]  ≥ Δ) //least-delay value exceeds delay bound
3.      stop routing execution
4.  else  // continue forward routing process
5.      if (μ not exhausted)
6.          if (b(e) + Req _B < CB) and (dly_so_far ≤  Δ) // CAC test
7.              send Call_Setup_Req (Req _B, Δ, μ, d(next), path_info) to next
8.              // next is determined according to routing schemes i.e., DCUR, DCR or DDCR
9.      else
10.         discard message
11.     end if
12. end if
```

Figure 4.3a Algorithm for source node during forward routing stage

at the time of ACK message arrival at the source node then resources will be allocated on the way back from destination towards source node, establishing a connection. Otherwise NACK message (line 18) will be send along the reserved path fragment to indicate the failure of the path (lines 22-24).

```
13. ACK (Call_ID, Req_B, block_links, path_info) received from node v on link e
14. if (μ not exhausted)
15.     if (b(e) + Req_B < CB) and (dly_so_far ≤  Δ) // CAC test
16.         Reserve Resources on e  // establishing connection
17.     else
18.         send NACK (Call_ID, path_info, block_links) to node v   // call failure
19.     end if
20. end if
21. end ACK
22. NACK (Call_ID, path_info, block_links)
23.     remove path info and release resources
24. end NACK
```

Figure 4.3b: Algorithm for source node during backward setup stage

4.2.3.2 Algorithm for Intermediate Node

Pseudo-code descriptions for the intermediate node in any reachable path during forward routing and backward (connection) setup stage are shown in Figure 4.4a and 4.4b respectively. Figure 4.4a shows an intermediate node at forward routing stage. CAC test will be performed on every out going link, which is determined by our selected routing schemes, i.e., DCUR, DCR and DDCA, such that the Call_Setup_Req message will proceed only if the setup time limit is not exhausted and the local node has the required resources. Otherwise, the Call_Setup_Req message will be discarded (line 13).

```
1.  Call_Setup_Req (Req_B, Δ, μ, dly_so_far, path_info) received from node v on link e
2.  if (μ is not exhausted)
3.  e = outgoing link //selected by scheme DCUR, DCR or DDCA
4.      if (b(e)+Req_B < CB) and (dly_so_far ≤ Δ) //CAC test
5.          send Call_Setup_Req (Req_B, Δ, μ, dly_so_far + d(next), path_info) on link e
6.          if current_node == destination_node //enter backward setup process
7.              send ACK (Call_ID, Req_B, block_links, path_info) from node v on link e
8.          else
9.              send Call_Setup_Req (Req_B, Δ, μ, dly_so_far + d(next), path_info) on
                link e
10.         end if
11.     end if
12. else
13.     discard message
14. end if
15. end Call_Setup_Req
```

Figure 4.4a: Algorithm for intermediate node during forward routing stage

If the CAC test is successful (line 4) and the Call_Setup_Req reaches a destination node, it will enter the backward setup (connection configuration) stage (line 6). Each time a Call_Setup_Req message (line 5) reaches the destination, the path information it carries will be stored in the destination database. An ACK message will be send along the path in order to establish a connection between sending and receiving ends (line 7). Otherwise forward routing will continue and a Call_Setup_Req message that carries the call QoS requirements and the shortest path QoS parameters, the accumulated delay so far till the current node will be forwarded on link e to next node (see line 9).

In Figure 4.4b, lines 16-24 depict the case where Call_Setup_Req is received at an intermediate node during backward setup stage. If setup time limit has not yet expired (line 17) then the node checks at line 18 if the QoS requirements are met. The routing process continues by sending an ACK message, reserving resources and establishing a connection (lines 19-20 and lines 25-28). Meanwhile Call_Setup_Req are forwarded (line 21) to the next node on the path after updating QoS parameters, i.e., accumulative delay with that of the next link to be traversed. On the other hand, if reservation fails at an intermediate node due to CAC conditions, i.e., required bandwidth and delay bound are not met, a NACK message (line 30) is sent on the path and all the way to the destination. Every node that receives the NACK message (lines 30-35) will remove the path information from the routing table and release the previously reserved resources for the connection. When the NACK message reaches the destination, it will trigger the destination (see Figure 4.5b) to choose the next best feasible path from its database. The NACK message is assumed to carry a list of all blocked links encountered in trying to setup the corresponding connection on the previous path. The blocked list will be used when selecting the next viable path to avoid previously blocked links.

```
16. Call_Setup_Req (Req_B, $\Delta$, $\mu$, dly_so_far, path_info) received from node v on link e
17. if ($\mu$ is not exhausted)
18.     if ( b(e)+ Req_B < CB) and (dly_so_far $\le \Delta$) //CAC test
19.         send ACK (Call_ID, Req_B, block_links, path_info) to next on link e
20.         // connecting link e to nexte= link connecting next
21.         send Call_Setup_Req (Req_B, $\Delta$, $\mu$, dly_so_far + d(next), path_info) to next
22.     end if
23. end if
24. end Call_Setup_Req
25. ACK (Call_ID, Req_B, block_links, path_info) received node v on link e
26. if ($\mu$ is not exhausted)
27.     if ((b(e)+REQ_B  < CB )
28.         Reserve Resources on e  //establishing a connection
29.     else
30.         send NACK (Call_ID, path_info, block_links) to node v // call failure
31. end if
32. end ACK
33. NACK (Call_ID, path_info, block_links) to node v
34.     remove path info and release resources
35. end NACK
```

Figure 4.4b: Algorithm for intermediate node during backward setup stage

4.2.3.3 Algorithms for Destination Node

A pseudo code of the algorithm for the destination node at forward routing and backward setup stages are shown in Figures 4.5a, and 4.5b, respectively. In Figure 4.5a, lines 1-5 shows completion of the routing stage. The destination node stores all the path information in its database and waits for a specific time period τ before it enters a backward setup stage. The philosophy behind wait time period at the destination node is to assure that all path information has reached and been stored at the destination database. At the same time it also ensures that none of the paths that have been found are omitted from consideration as viable path before the destination node selects the viable path and enters a backward setup stage.

```
1.  Call_Setup_Req (Req_B, Δ, μ, dly_so_far, path_info) received from node v on link e
2.  if (μ is not exhausted)  and current == destination  // d has reached
3.      wait until τ is reached
        // d stores path info in its database and enters backward setup stage
4.  end if
5.  end Call_Setup_Req
```

Figure 4.5a: Algorithm for destination node during forward routing stage

```
6.  NACK (Call_ID, path_info, block_links) receives from node v
7.      remove path info and release resources
8.      if (μ is not exhausted) and (τ exhausted) // d has reached, wait until τ is reached
9.          select best viable path
10.         send Call_Setup_Req (Req_B, Δ, μ, dly_so_far, path_info) on link e
11.     end if
12  end NACK
13. Call_Setup_Req (Req_B, Δ, μ, dly_so_far, path_info) received from node v on e
14.     if ( b(e) + Req_B < CB) and (dly_so_far ≤ Δ) //CAC test
15.         send ACK (Call_ID, Req_B, block_links, path_info) to next on link e
            // connecting link e to next
16.         send Call_Setup_Req (Req_B, Δ, μ, dly_so_far + d(next), path_info) to next
17.     end if
18. end Call_Setup_Req
19. ACK (Call_ID, Req_B, block_links, path_info) received from node v on link e
20. if (b(e)+Req_B  < CB ) and (μ is not exhausted) //CAC test
21.     Reserve Resources on e  //establishing a connection
22. else
23.     remove call information from database
24.     send NACK (Call_ID, path_info, block_links) to v // call failure
25. end if
26. end ACK
```

Figure 4.5b: Algorithm for destination node during backward setup stage

Figure 4.5b shows the destination node at the backward setup stage. When the NACK message reaches the destination from node ***v***, it will trigger the destination to choose the next best feasible path from its database (lines 6-9), after a wait time period and if the call setup time is not exhausted. A Call_Setup_Req will be send out along the chosen viable path (line 10).

As the first Call_Setup_Req message (lines 13-18) is received at the destination, the connection configuration stage starts on the already chosen feasible path from the set of paths from the destination database (line 9). The destination chooses a least cost path meeting the delay bound for establishing a connection. Then, it sends an ACK message (line 15), which carries the selected path information, backward on the chosen path to start reserving the required resources. Upon receiving the ACK message (19-20), the CAC criteria will be tested. Reservation continues (line 21) on the selected path until a source node is reached, where at this point all links on the path have been reserved and the connection has been established successfully. The connection configuration process will continue until one of three things happen; the destination succeeds to connect to the source node, there are no more viable paths available at the destination database, or the setup time limit is exhausted and the connection is timed out. The reservation process is completed once the ACK message reaches the source node connecting the destination on a unicast path. The destination on the other hand, will know about the connection establishment when it receives the first data packet. In case of CAC test violation, all call information from database will be removed (line 23) and NACK message (line 24) will be sent out.

4.3 Summary

This chapter introduced our proposed new approach of Integrated Routing Protocol (IRP). IRP establishes real-time connection using forward routing and backward setup approach. During the forward routing process, feasible paths are found and during the backward connection setup process sufficient network resources such as network bandwidth, buffer space, etc. are reserved at each network node for the connection so that the required quality of service can be guaranteed. If a call request is rejected when setup time limit expires for a session while allocating resources backwards from the destination towards source node, the partially (or fully) constructed path is torn down, and the already allocated resources so far will be released and made available for other connections. Cost and delay are considered as the routing metrics. Cost is taken as a function for efficiently measuring network resource (bandwidth) utilization, while delay is taken as the queuing delay and propagation delay along the path experiences.

Chapter 5

Performance Evaluation

The performance of IRP algorithm is analyzed and compared with the performance of the individual implementations of DCUR, DCR and DDCA algorithms. A discrete-event base simulator is developed using the Visual C++ programming language. The simulator randomly generates a network graph over a square grid area using average node degree of four. Background network traffic is randomly generated on edges to show varying real life networking conditions. Information on network nodes (routers) is periodically exchanged among neighboring nodes, using the Distance-Vector protocol. The performance evaluation depends on the accuracy of network state information that changes frequently due to dynamic nature of distributed routing. Controlling how often the updates are sent could reduce the volume of information updates. Yilmaz and Mata [YiM02] suggest updates can be sent periodically, or triggered by a change that is bigger than a specified value or clamp down timers. Furthermore, choosing what to advertise can also control the volume of updates. For instance, can we advertise information only in the link of a node that triggered the update or all the links of the node in a graph.

The rest of this chapter is organized as follows. Section 5.1 explains the simulation model adopted in this study. Section 5.2 describes the experimental settings used in this simulation. Section 5.3 describes the performance evaluation metrics that are taken into consideration, which includes call acceptance ratio (CAR), call blocking ratio due to lack of bandwidth (CBRLB) and call blocking ratio due to setup time expiry (CBRSTE). The effects of network traffic load, connection required resources, networks density and call setup time limit on the performance of distributed destination-controlled IRP scheme are investigated and reported in Section 5.4. Section 5.5 concludes this chapter.

5.1 Simulation Model

The simulation model consists of two components: the network model and the background traffic mode.

5.1.1 Network Model

The simulations are based on network topologies generated using a modified version of the Waxman Model [Wax98] as proposed by Salama [Sal96]. Networks of different densities (nodes 20 and 30) are randomly generated over a square coordinate grid area, generating uniformly distributed values for x and y coordinates in a plane. Edges are introduced between each pair of nodes with probability **P(u,v)** that depends on distance between nodes u and v**.** The distance between nodes **u** and **v** is computed through well-known Euclidean metric, denoted as **dist (u, v)**. The edge probability is defined as: $\mathbf{P(u,v) = (\beta * \exp(-dist\ (u,v))/(\alpha * L))}$, where L is maximum distance between nodes α and,β are tunable parameters in the range (0,1]. The parameter α controls the density of short edges in the network such that increasing α increases the number of connections between far away nodes. The parameter β controls the average node degree, such that increasing β increases the degree of each node. A random number, r, is generated such that $\mathbf{0 \leq r < 1}$**.** If **r** is less than **P(u,v)** then an edge is added between nodes. A connected digraph is created with an average node degree of four. The edge cost and edge delay are uniformly distributed within the specified range. The edge delay is made proportional to the Euclidean distance of the edges in the coordinate plane.

5.1.2 Background Network Traffic Model

The background network traffic is modeled in the simulation by assigning a value for the reserved bandwidth on links. The reserved bandwidth on the link reflects the network load, such that higher reserved bandwidth values represent higher network loads. The reserved bandwidth comprises of three values, i.e. minimum, maximum and the nominal assigned bandwidths. Each link in the network is randomly assigned any of these three values. The nominal assigned bandwidth on a link serves the purpose of the network load of our interest. When the network graph is generated, all the links in the entire graph are flooded with nominal assigned bandwidth as the desired network load. This is assumed to be background traffic and is randomly updated to reflect changes in link loads. A change in a link load updates the cost of this link, the buffering delay and ultimately resulting in updating the link delay. Traffic updates are periodically advertised to immediate neighbors using a distributed version of Bellman-Ford distance vector protocol.

5.2 Experimental Settings

All simulation experiments conducted are divided into two sets in order to simulate networks with different densities. The first set of experiments is for networks of 20 nodes and the second set consists of a network of 30 nodes. The simulation results are achieved with 10% confidence intervals and 90% confidence level (see Appendix A).

Each simulation result is observed as an arithmetic average of five consecutive simulation runs. Each run consists of 4000 requested calls. For each run, a new graph is generated with a different topology using different values for α and β. Source and destination nodes are picked randomly from the set of randomly generated nodes. Each link e is assigned a bandwidth (cost) and a delay parameter. The bandwidth parameter is associated with the available bandwidth on the link, such that the higher available bandwidth reflects higher value. The delay parameter is the sum of the perceived buffering delay, transmission delay and propagation delay over the link [Rou00] and [RoH01].

In IRP the destination node collects arriving setup messages within a route selection waiting time period (τ), which is a predefined timer for selecting the best-cost path during forward routing stage. After the waiting time period elapses before selecting viable path, the destination node releases the ACK message. The backward stage starts with an ACK message that is forwarded backward towards the source node along the best-cost selected path. If a link on the selected path breaks due to either of three reasons, i.e., delay violation, lack of resources availability or setup time expiry, the ACK message is discarded and a negative ACK message i.e., NACK is sent backward along the path fragment to the destination. The destination node will then choose another path, which does not contain any previously broken links. When the source node receives an ACK message, it knows that a path has been established to the destination and then start transmission. On the other hand, the pure implementation of DCUR, DCR and DDCA algorithms need not to wait before releasing the ACK message. This is due to the fact that these algorithms generate a single path for a pair of source and destination node. Since there is no chance of an alternate path, the ACK message is rather released immediately by the destination node after it receives the first Call_Setup_Req message.

5.3 Performance Evaluation Metrics

The three network performance metrics studied in this research are:

The Call Acceptance Ratio (CAR): CAR represents the proportion of calls accepted (a feasible path connecting the sending and receiving ends is found within a setup time limit, μ).

The Call Blocking Ratio due to Lack of Bandwidth (CBRLB): During the connection setup process, sufficient network resources (bandwidth) are reserved at each network node so that the user required quality of service could be guaranteed for data transmission time. However, in case of a failure in finding the desired resources within the traversed path, a call is blocked due to lack of resources (bandwidth) and is called CBRLB.

The Call Blocking Ratio due to Setup Time Expiry (CBRSTE): A call is blocked due to setup time expiry if a call request fails to establish a connection within μ. The connection is said to have timed out when μ expires during a session, while allocating resources. The partially (or fully) constructed path will be torn down and already allocated resources will be released which will be made available for other connections.

5.4 Simulation Results

We examined the effect of network load, connection bandwidth requirement, delay bounds, waiting time period and call setup time limit on the performance of the proposed IRP scheme. The simulation results obtained for IRP are analyzed and compared with the results of DCUR, DCR and DDCA algorithms. Various experiments are conducted under different network traffic conditions to evaluate the performance measures of interest, i.e., call acceptance ratio (CAR), call blocking ratio due to lack of bandwidth (CBRLB) and call blocking ratio due to setup time expiry (CBRSTE). It is expected that IRP, which chooses the best path chosen by the DCUR, DCR and DDCA algorithms should have a higher CAR and lower CBRLB than any of the three routing algorithms.

The first experiment studies the CAR for IRP and compares it with the CAR of DCUR, DCR and DDCA algorithms for lenient delay bounds (Figure 5.1) and strict delay bounds (Figure 5.2). The second experiment shows the CBRLB for lenient delay bounds (Figure 5.3) as well as tight delay bounds (Figure 5.4). The third and fourth experiments were conducted to study the effect of CAR (Figures 5.5 and 5.6) and CBRLB (Figures 5.7 and 5.8) for all four algorithms under a single plot for light and heavy bandwidth requirement, as well as for both lenient and tight delay bounds using varying traffic conditions. The results on the left hand side in all the plots are the results obtained at network of size 20 nodes while that on the right hand side are the results obtained at network of size 30 nodes. The call setup time limit in these first four experiments was taken very large, so that it has no effect on call blocking probabilities. However, we studied the effect of call setup time limit in the last two experiments for IRP. Results for these experiments are shown in Figure 5.9 and Figure 5.10.

Figure 5.1 shows the CAR for DCUR, DCR, DDCR and IRP for four values of required bandwidth (Req. Band values of 15, 20, 25 and 30) under a lenient delay bound of 200 ms. Figure 5.1a shows the CAR for DCUR scheme at relaxed delay bound for networks of 20 nodes. When the connection required bandwidth is light (Req-Band =15) the CAR is 90% and decreases slowly with the increase in network load. At medium connection requirement (Req. Band 20 and 25) the CAR at light loads is over 80% but gradually decreases with the increase in network load, whereas, at very high load, CAR abruptly lowers down to 50 %. For high required bandwidth (Req. Band =30), the CAR is below 70% for lighter loads but as the load increases, feasible paths with high bandwidth requirement gets harder to find, hence resulting in lower call acceptance ratios. Figure 5.1b and Figure 5.1c show the CAR for DCR and DDCA schemes, respectively. It is evident from the results in these figures that call acceptance deceases for higher network loads particularly when the connection QoS requirement is high. This is because it is hard for these algorithms to find least-cost paths that meet the delay constraint. On the other hand, the IRP (Figure 5.1d) shows 95% call acceptance at lighter load and low connection requirement. It achieves a significant call acceptance of 75% at light load with high connection requirement. The performance of IRP remains stable over 55% and does not degrade abruptly at high network loads.

Network size also affects the call acceptance ratio. As the network size grows, the propagation delay between end nodes increases. Finding the desired delay constraint paths in dense networks reduces the probability of call acceptance in DCUR, DCR and DDCA schemes (Figures 5.1e-5.1g).

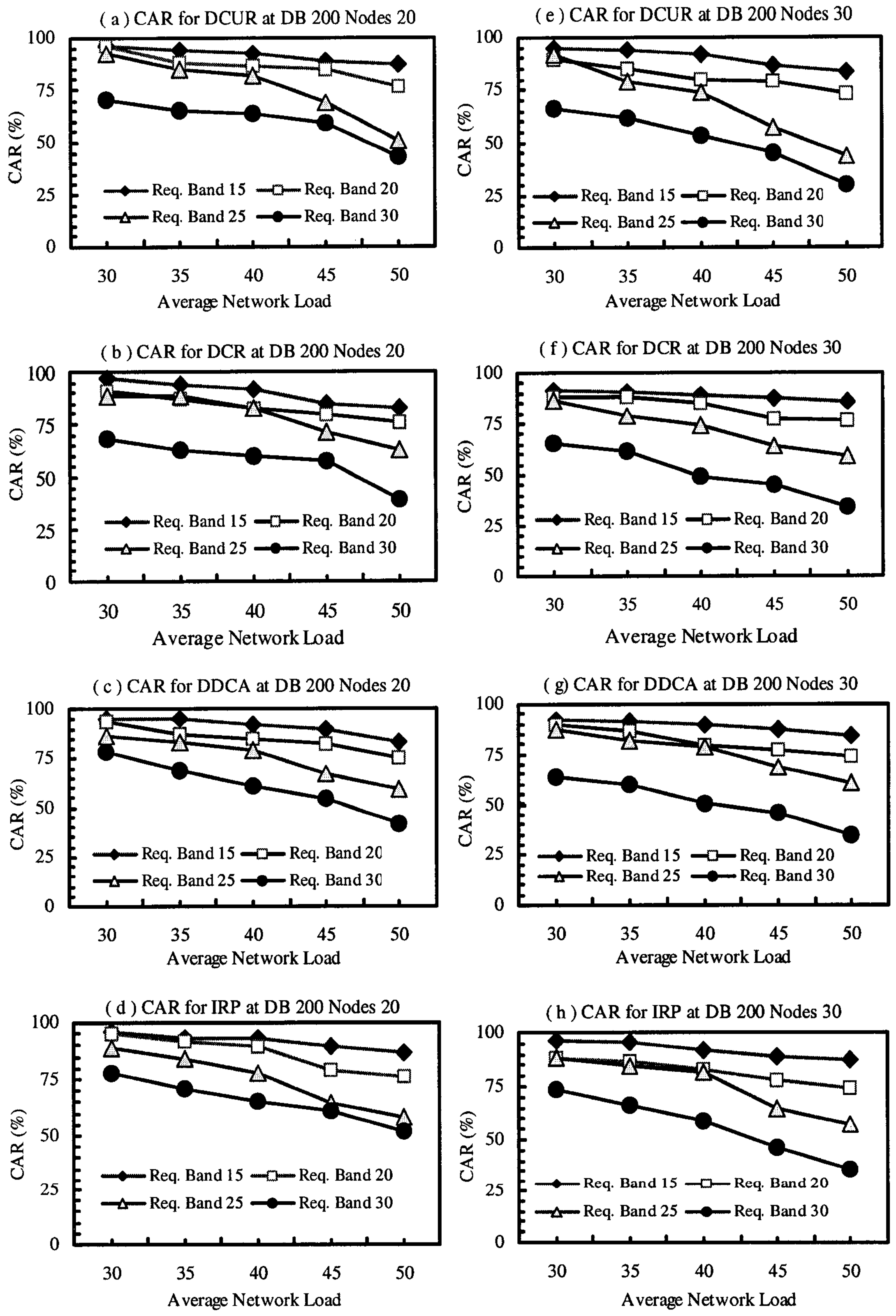


Figure 5.1 Call Acceptance Ratio vs. Network Load at Delay Bound 200

However, IRP performs better (Figure 5.1h) comparing to counterpart schemes due to the fact that IRP provides alternate path choice, henceforth resulting in higher call acceptance even in dense networks. The proposed IRP algorithm clearly outperforms DCUR, DCR and DDCA algorithms in call acceptance probabilities for all four values of required bandwidths at various network densities, traffic load when the delay bound is lenient.

Figure 5.2 shows CAR for all the schemes at tight delay bound. Comparing Figures 5.1a, 5.1b and 5.1c with Figures 5.2a, 5.2 b and 5.2c reveals that as the delay bound increases, achieving higher CAR becomes harder in all the three schemes. This is because most of the calls are blocked due to delay constraint violation. Whereas, IRP yields higher CAR of 95% for both relaxed (Figure 5.1d) and 90% for tight delay bounds (Figure 5.2d) at low connection requirement than DCUR, DCR and DDCA (Figures 5.1a-5.1c and Figures 5.2a-5.2c) for small networks (nodes 20). As the network size increases, in case of tight delay bounds (Figures 5.2e-5.2g), the CAR starts degrading significantly for heavy connection requirement as the network loads increases for all the three schemes. However, IRP still shows better results (over 20% CAR) comparing to DDCA (Figure 5.2g) where CAR is below 10%, while DCUR and DCR shows better results (near 20 % CAR) than DDCA. The low CAR in these schemes is due to high delay violation, as dense networks experiences longer propagation delays that result in higher delay violations and low call acceptance ratios.

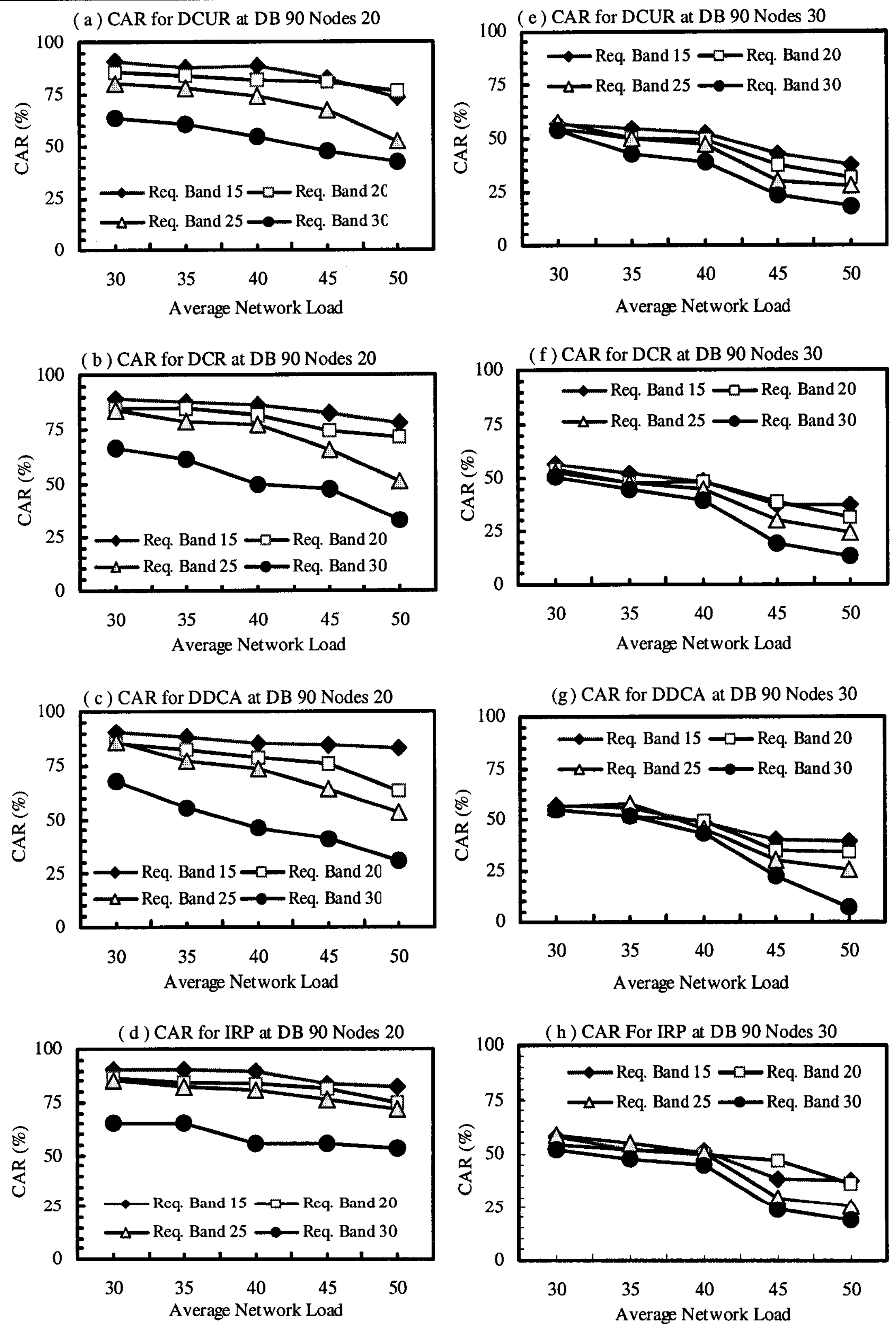


Figure 5.2 Call Acceptance Ratio vs. Network Load at Delay Bound 90

The second experiment studies the call blocking probability due to lack of resources. Higher traffic loads yield higher CBRLB, at the same time, heavier connection QoS requirement yields higher CBRLB too. Figure 5.3 and Figure 5.4 show the CBRLB for all schemes at relaxed as well as tight delay values respectively. In Figures 5.3a-5.3d all schemes including IRP show similar low trends of CBRLB at low bandwidth requirement (under 1 %) which has little effect on call acceptance, suggesting that all the schemes easily find feasible paths that have low resource requirements. As the connection requirement and network load increase the CBRLB for DCUR, DCR and DDCA show increasing trends (Figures 5.3a-5.3d), whereas IRP (Figure 5.3d) achieves low CBRLB both at light as well as heavy connection requirements. As it can been seen that at 30 % network load, in IRP only 20% calls are blocked due to lack of resources when a high connection requirement is needed (Figure 5.3d) while DCUR (Figure 5.3a) and DDCA (Figure 5.3c) reject 25% of calls and DCR (Figure 5.3b) rejects 30% of calls due to lack of recourses. As the network load increase from 30 % to 50 %, IRP still maintains low CBRLB (45 %), whereas DCUR, DCR and DDCA show over 50 % CBRLB.

Clearly, IRP outperforms DCUR, DCR and DDCA in achieving low call blocking ratios both at high and low connection requirements as well as at light and heavily congested network loads. IRP outperforms the counterpart algorithms in dense networks too. As it can be seen from Figures 5.3e-5.3g at low bandwidth requirements the CBRLB tends to increase up to 5% for high network loads and at heavy bandwidth requirement CBRLB increases up to 55 % due to the fact DCUR, DCR and DDCA need more resources to establish successful connections. On the contrary, IRP (Figure 5.3h) shows more than 10% lower CBRLB at heavy connection requirements and high network load.

Figures 5.4a-d and 5.4e-g shows CBRLB for DCUR, DCR and DDCA algorithms for small and large networks respectively for tight delay bound. It is observed that the delay constraint has little effect on call blocking ratio due to lack of bandwidth for small networks (but networks density has a significant effect on CAR). In large networks due to longer propagation delays, the link delay increases and hence more calls are blocked due to delay violation rather than lack of resource availability, resulting in low CAR. Nevertheless, DCUR, DCR and DDCA result in low CBRLB (Figures 5.4e-5.4h) as compared to call blocking at relaxed delay (Figures 5.3e-5.3h). IRP achieves lower CBRLB for nodes 30 as well as nodes 20 networks (Figures 5.3d, 5.3h and Figures 5.4d, 5.4h) compared to other schemes (Figures 5.3a-5.3c and Figures 5.4e-5.4g). IRP has the advantage of alternate paths availability over DCUR, DCR and DDCA algorithms, resulting in higher CAR and ultimately lower CBRLB.

Figure 5.5 and Figure 5.6 show the call acceptance ratio versus network load under light and heavy bandwidth requirement, respectively. The figures combine the results of IRP with DCUR, DCR and DDCA algorithms for both light and heavy bandwidth requirement at various network loads.

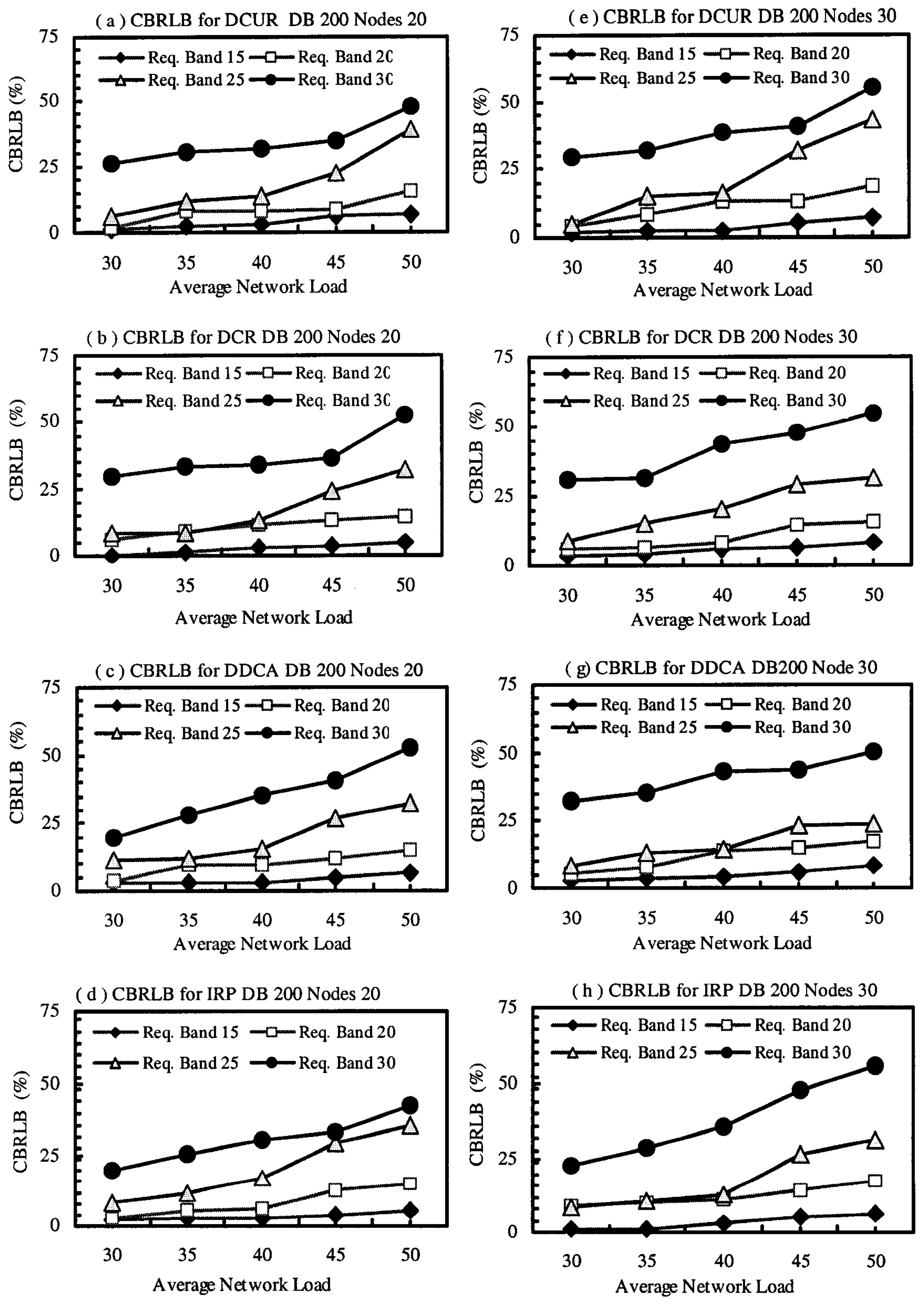
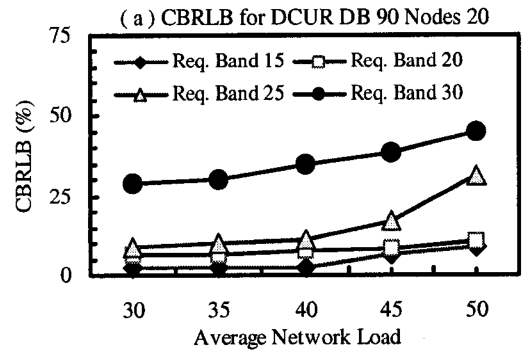
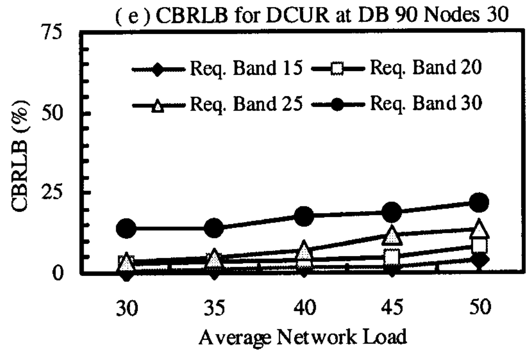
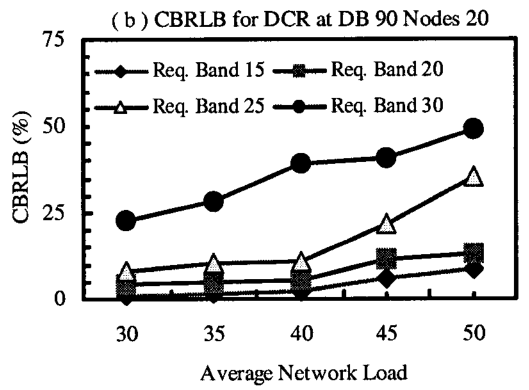
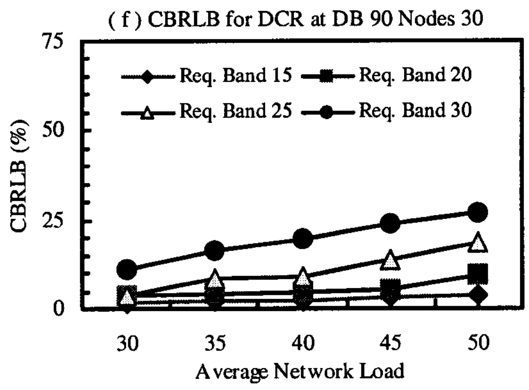


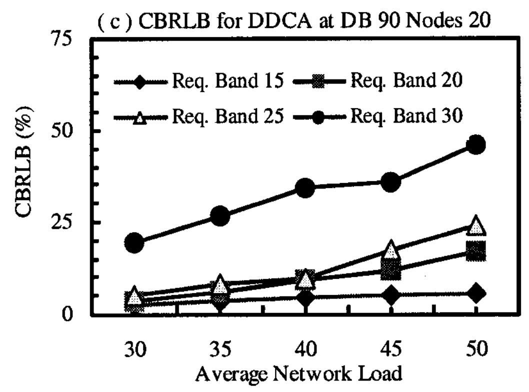
Figure 5.3 Call Blocking Ratio vs. Network Load at Delay Bound 200

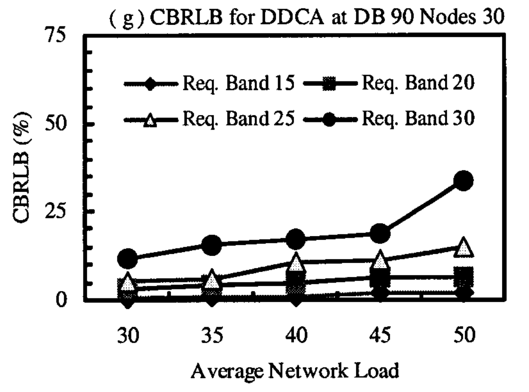


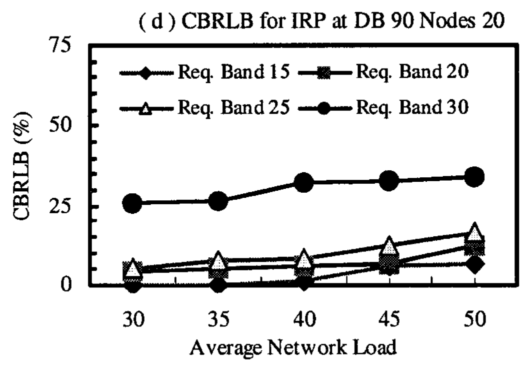












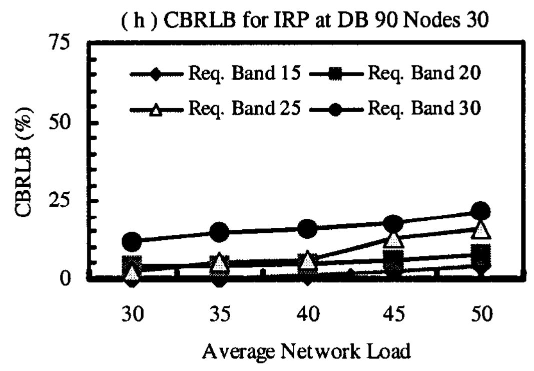


Figure 5.4 Call Blocking Ratio vs. Network Load at Delay Bound 90

Networks of 20 and 30 nodes were generated for both lenient and tight delay bounds to study CAR and CBRLB results. In Figure 5.5, the experiments are conducted under a relaxed delay bound to reduce its effect on the call acceptance ratio and another time to elucidate under a tight delay bound (Figure 5.6). As the network load increases while the delay bound is relaxed, the call acceptance ratio decreases slowly for all routing algorithms, which can be attributed to the fact that it is harder to find links with the required QoS resources in highly congested traffic conditions. The network size also affects the call acceptance ratio. With large networks, meeting a delay bound is harder due to longer propagation delays. Therefore as the network size increases, the distance between end nodes (propagation delay) increases and results in longer delays. Ultimately finding paths with least-cost links that meet the delay bounds becomes harder to find.

As the delay bound is assumed to be tighter (Figure 5.6b and 5.6d), the performance of DDCA decreases faster than IRP algorithm. The delay bound starts to have a major effect on call acceptance ratio, which increases the difficulty of finding a path that meets both the bandwidth and the delay requirement. This particularly is true in case of large networks (Figures 5.6c and 5.6d). Nevertheless, IRP still outperforms the other three algorithms at lighter loads.

IRP performs better in terms of call acceptance than DCUR, DCR and DDCA algorithms because IRP provides alternate path option between end nodes. It is observed from the results in Figures 5.7a-5.7d that IRP adopts well to call blocking probabilities and results in high CAR. The lower CBRLB in IRP is the result of up- to-date information carried in

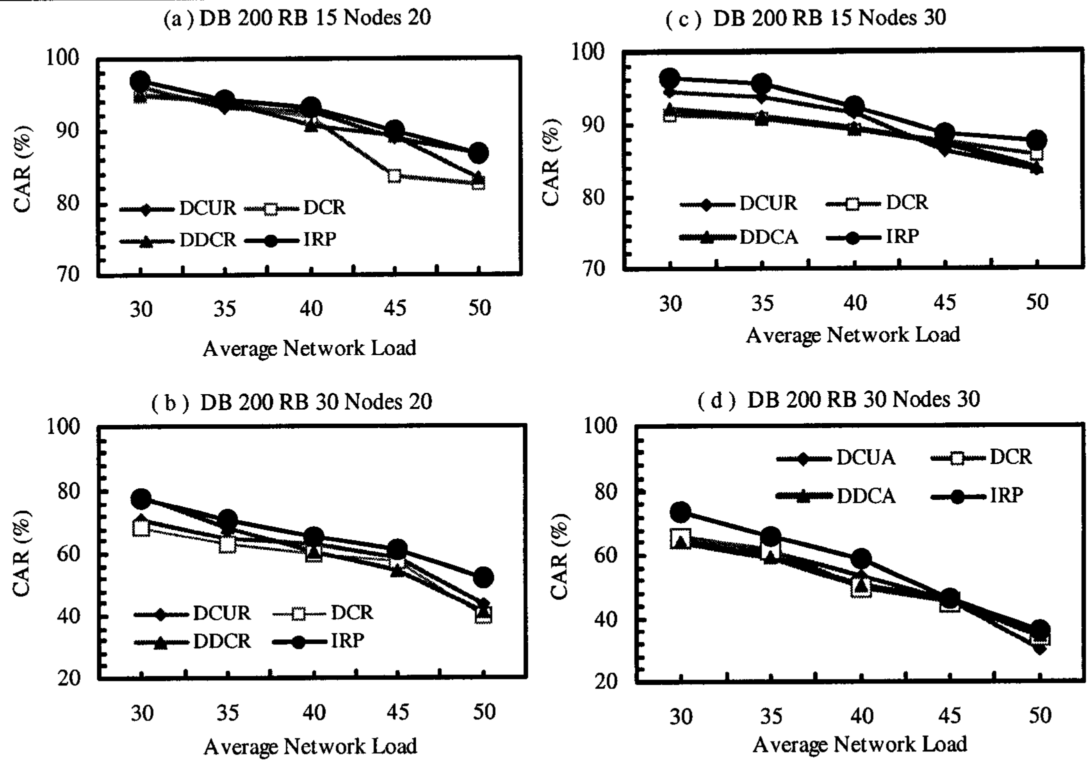


Figure 5.5 Call Acceptance Ratio vs. Network Load at Delay Bound 200

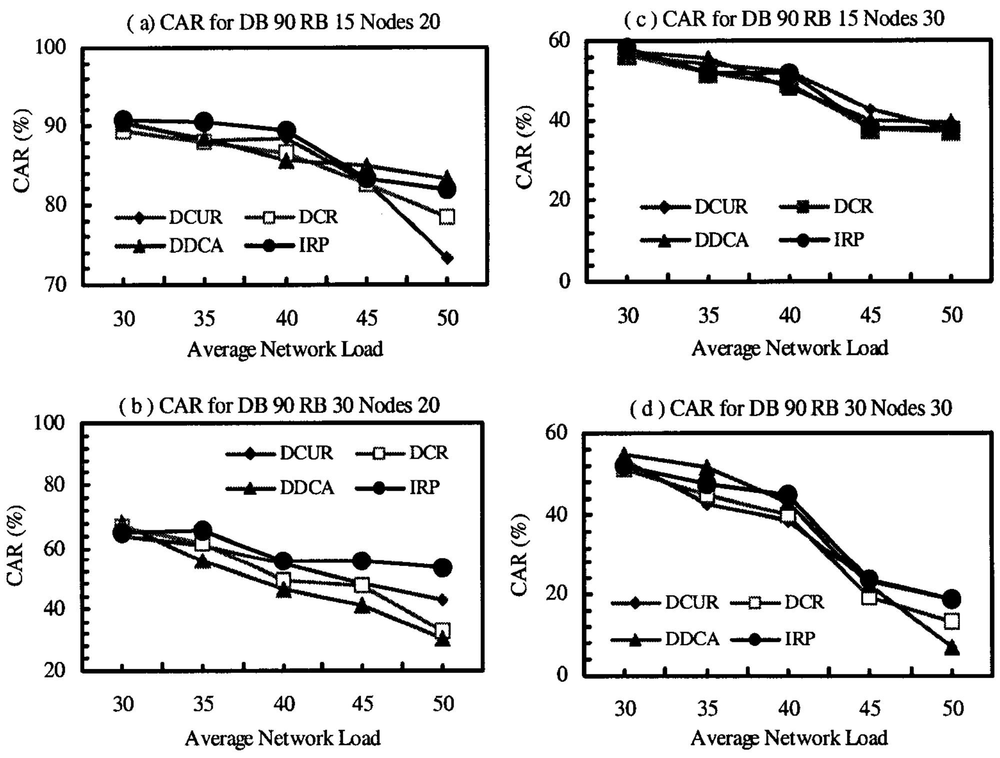


Figure 5.6 Call Acceptance Ratio vs. Network Load at Delay Bound 90

the Call_Setup_Req messages to the destination and also the ability of IRP to choose an alternative path in case of resource reservation failure within traversed paths. When the network load gets extremely high, the call acceptance ratio by DCUR, DCR and DDCA decreases faster than IRP, as it excludes all links with insufficient bandwidth from consideration when computing a shortest path.

For lenient delay bound, as in Figure 5.5a and 5.5c, the call acceptance for DCUR, DCR and DDCA degrades for both light and heavy bandwidth requirement when the number of network nodes increases. Whereas for tighter delay bounds and heavy bandwidth requirement, as in Figure 5.6b and 5.6d, the call acceptance degrades faster (drops down below 10%) in all the three schemes, whereas IRP maintains above 50 % CAR in small networks and over 20% of CAR in dense networks. This shows that if a connection fails on a feasible path in case of any of three schemes the entire scheme would fail. This is not the case for IRP, hence the higher call blocking probabilities as shown in Figures 5.8a-5.8d.

Indeed, the alternate path choice is the reason IRP outperforms DCUR, DCR and DDCA algorithms in terms of call acceptance probability

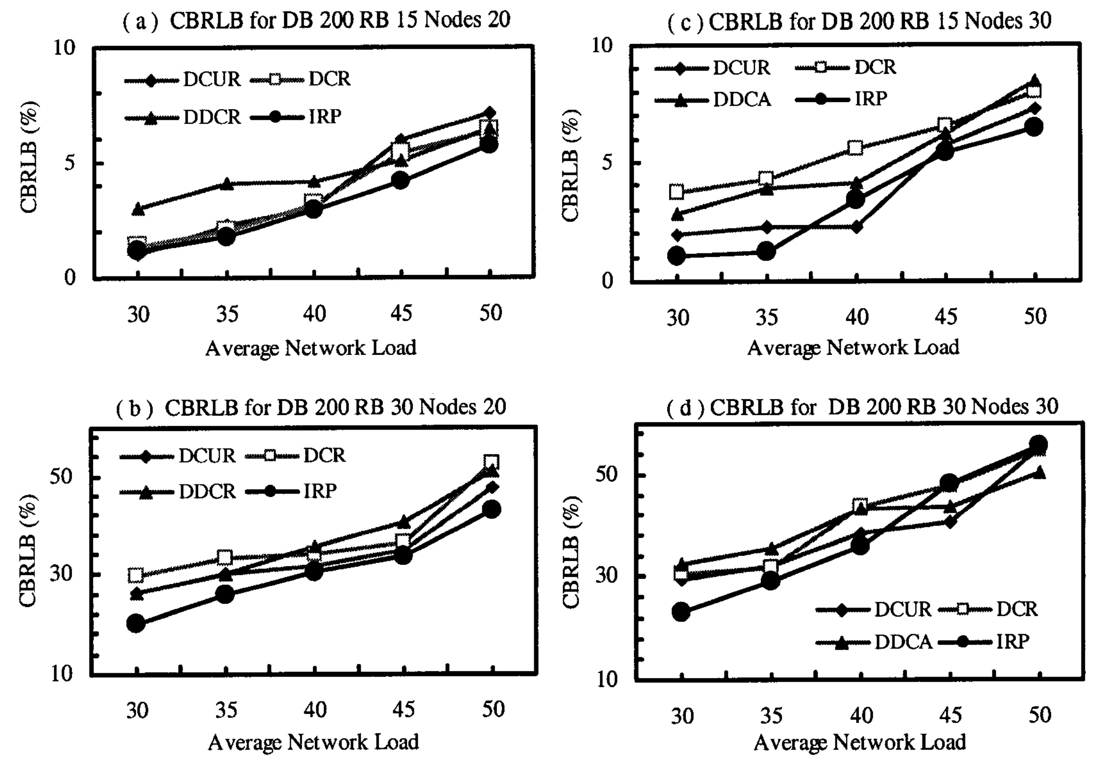


Figure 5.7 Call Blocking Ratio vs. Network Load at Delay Bound 200

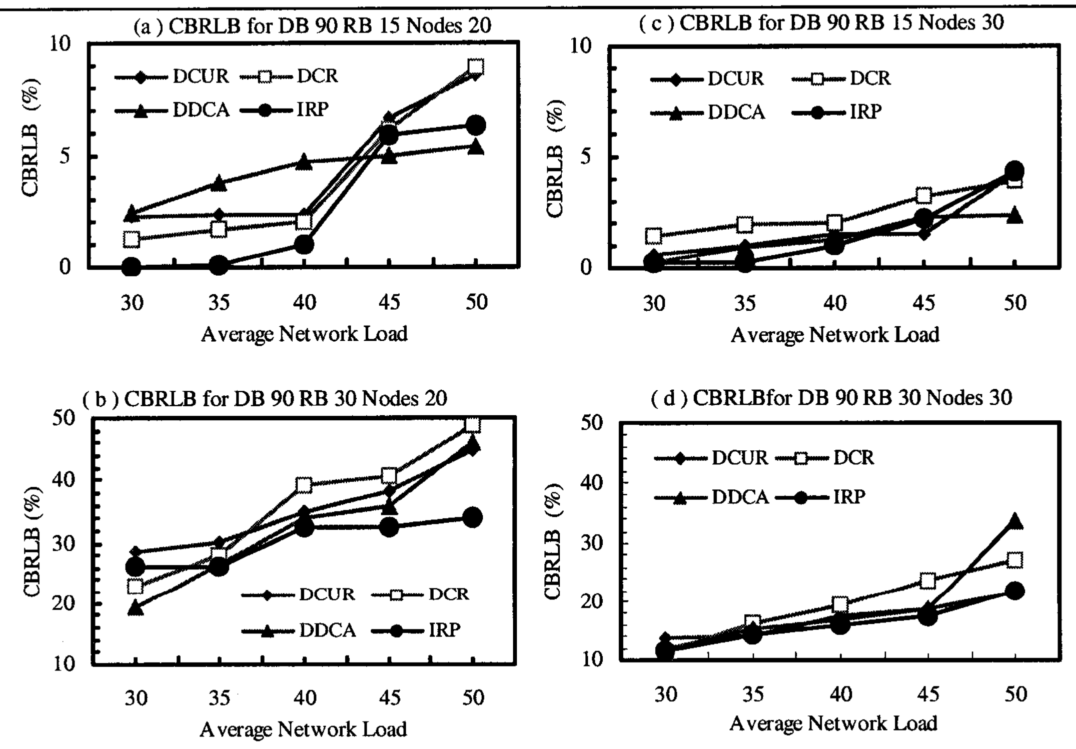


Figure 5.8 Call Blocking Ratio vs. Network Load at Delay Bound 90

The effect of call setup time limit, μ, is studied in case of IRP scheme under a tight delay bound as shown in Figures 5.9a and 5.9b. The call setup time limit has significant effect on call blocking probability. Figures 5.9a and 5.9b show the call blocking ratio due to setup time expiry (CBRSTE) for longer, moderate and tighter setup time limits in small and dense networks, respectively. For longer setup time limits (μ=150ms), none of the calls is blocked due to setup time expiry. Setup time μ = 85 ms yields lower CBRSTE in case of smaller networks, as is shown in Figure 5.9a. However, in dense networks CBRSTE is 40 % at light load for μ =85ms and decreases with the increase in network load as is shown in Figure 5.9b. It is because, at higher network loads, more calls are blocked due to delay violation and lack of resources rather than setup time expiry. When setup time limit is made tighter (μ = 50), 50% calls are timed out at smaller networks as is shown in Figure 5.9a. Since the setup limit is very short, the IRP is not able to take advantage of alternate path availability and go back to the destination to select the next viable path. Therefore, it results in high CBR due setup time expiry. Whereas, in dense networks at μ = 50, 40% calls may time out.

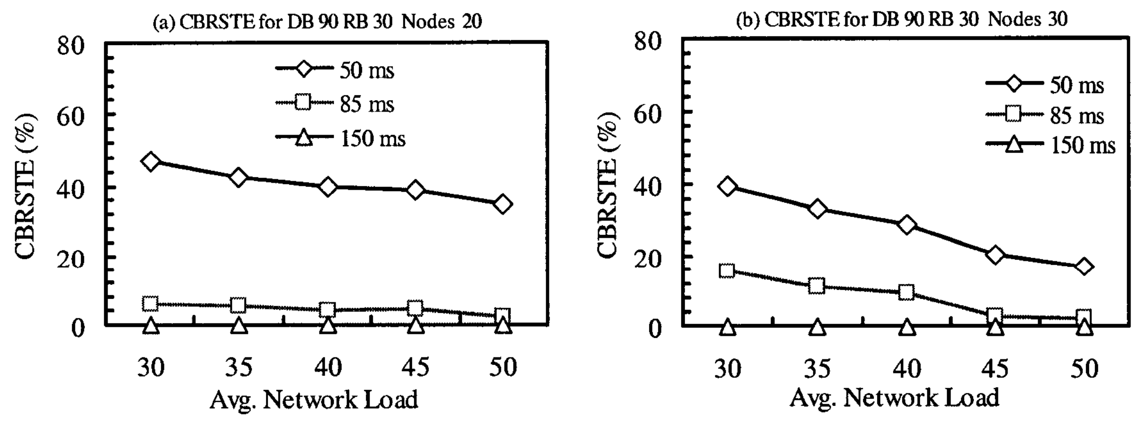


Figure 5.9 Call Blocking Ratio due to Setup Time Expiry vs. Network Load

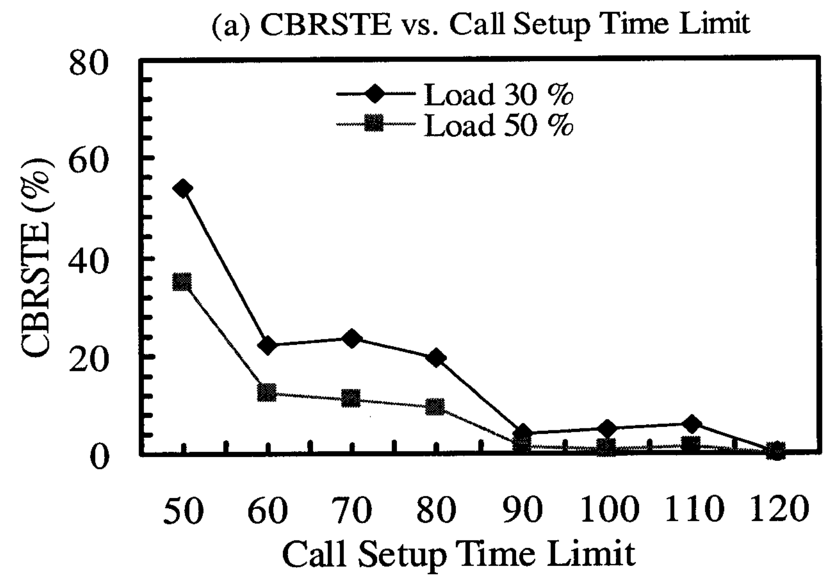


Figure 5.10 Call Blocking Ratio due to Setup Time Expiry vs. Call Setup Time Limit

Figure 5.10 shows CBRSTE versus call setup time limit (μ) under delay bound of 90 for low and high average network loads of 30 % and 50 % respectively. It is evident from the figure that as μ increase, the CBRSTE decreases, both under light and heavy loads. However, at heavy load, more calls are timed out. This is attributed to the fact that at very large setup time limits, IRP stabilizes and calls are accepted before they are timed out where calls are rejected mainly due to failure in finding a path that meets the bandwidth and delay constraints.

5.5 Summary

This chapter investigated the performance of the proposed Integrated Routing Protocol through an extensive simulation model. Several experiments were conducted to evaluate the performance of IRP and to compare it with DCUR, DCR and DDCA algorithms. The simulation results showed that IRP outperforms DCUR, DCR and DDCA in achieving high call acceptance. The ability of providing alternate path choices between end nodes enhances the call acceptance ratio in IRP. At the same time, IRP is shown to have good call blocking probability results as it carries up-to-date CAC information and select paths with highest available bandwidth under the delay bound, hence avoiding selecting congested paths that would ultimately end up in call blocking during resource reservation process.

Chapter 6

Conclusions

The nature of applications needed to be transmitted over the Internet has changed significantly during the past two decad. This has resulted in the development of today's high speed networking architecture, that supports wide ranges of communication-intensive real-time multimedia applications. A key issue facing the network service provider is that of providing QoS guarantees to users while utilizing network resources as efficiently as possible. Network resources typically include link bandwidth, buffer space, etc. The service provider has a number of controls using which network performance can be tuned. These includes routing, bandwidth allocation, call admission control (CAC) and resource reservation. Selecting appropriate paths, the routing function can efficiently allocate network bandwidth (resources). Bandwidth allocation refers to the service rate assigned to a source at a physical link. A higher service rate normally implies better service (lower delays) but lower efficiency. The CAC function controls the acceptance of calls into the network. A new call is admitted only if its QoS requirements can be met while meeting those of all existing calls in the network. A good CAC algorithm should result in the maximum possible utilization of resources. The challenge of providing QoS has been accepted by many researchers, and they proposed several QoS routing solutions. The main objective of QoS routing is to provide efficient utilization of network resources while satisfying QoS requirements for every admitted connection.

In this thesis we investigated various QoS routing schemes and proposed a novel approach of Integrated Routing Protocol (IRP) in IP networks, to meet multi-media applications requirements while achieving efficient network resource utilization. We interweave routing with CAC and resource reservation. IRP establishes a unicast connection in two stages, a forward routing and a backward setup stage to find the most viable path between source and destination nodes. The forward routing stage computes QoS paths using distributed delay-constrained unicast routing algorithms described in Delay-Constrained Unicast Routing (DCUR) proposed by Salama et al. [SRV97], Delay-Constrained Routing (DCR) proposed by Sun and Langendorfer, [SuL98] and Distributed Delay-Constrained Algorithm (DDCA) proposed by Zhang et al., [ZKM01]. The routing information is forwarded from source towards destination on feasible paths through connection setup messages. The setup messages include the latest CAC information on links within the traversed paths. CAC is used in routing to test if a path can support the flow, while maintaining the QoS of the existing connection. In the backward setup stage, the destination node selects a viable least-cost path with the most available bandwidth that meets the delay constraint and begins resource reservation backwards towards the source node, exercising CAC on each link along the path. CAC decisions are based on the current network traffic load. The proposed approach requires each router to maintain routing entries for each concerned routing metric (e.g., delay, cost, etc.).

A comprehensive simulation model was developed to study the performance of the proposed scheme. The results were compared with those of the independent implementation of each of the algorithms used. Various simulation experiments were conducted under different traffic characteristics and network parameters. Since IRP provides alternate path techniques in unicast routing, it maximizes the number of accepted connections, while providing satisfactory QoS to existing connections and treating all requested calls fairly. IRP integrates CAC, resources reservation with routing. The reason for this integration is that multimedia traffic has special QoS requirements that can only be met by appropriate admission control mechanisms, at the same time reserving resources for admitted connections in advance is essential in order to guarantee the required services. Integration of routing with CAC, while reserving resources in advance can result in higher call acceptance.

References

[AAP93] B. Awerbuch, Y. Azar and S. Plotkin, "Throughput-competitive On-Line Routing," Proceedings 34th IEEE Annual Symposium on Foundations of Computer Science, Nov. 1993, pp. 32-40.

[AlO98] M. Al-Otaibi, "Routing and Resource Reservation in ATM Networks," MSc. Thesis, Department of Computer Science, Kuwait University, Kuwait, 1998.

[Auk96] Aukia, P, Oy, N., "Quality of Service Based Routing", http://www.tcm.hut.fi/Opinnot/Tik-110.551/1996/qos_rout.html

[BBB01] H. Bettahar, A. Bouabdallah, C. Beaujean, "A parameterized distributed unicast routing algorithm with end-to-end delay constraint," Proceedings ICT2001-The 8th IEEE International conference on Telecommunications-Romania, 2001, pp.1-16.

[ChN98a] S. Chen, and K. Nahrstedt, "An overview of quality-of-service routing for the next generation high-speed networks: Problems and Solutions," IEEE Network, Special Issue on Transmission and Distribution of Digital Video, Vol. 12, 1998, pp. 64-79.

[ChN98b] S. Chen, and K. Nahrstedt, "Distributed QoS routing with imprecise state information," Proceeding of IEEE Seventh International Conference on Computer, Communications and Networks, Lafayette, LA, Oct. 1998, pp. 614-621.

[ChN98c] S. Chen, and K. Nahrstedt,"Distributed QoS routing in High-Speed Networks Based on Selective Probing," Technical Report, University of Illinios at Urbana-Champaign, Department of Computer Science, 1998.

[GSA01] D. Ghosh, V. Sarangan, and R. Acharya, "Quality-of-Service Routing in IP Networks," IEEE Transactions on Multimedia, Vol. 3, No. 2, 2001, pp. 200-208.

[FeG00] A. Fei, and M. Gerla, "Smart Forwarding Technique for Routing with Multiple QoS Constraints," IEEE Proceedings of Globecom 2000, San Francisco, CA, Vol.1, Nov. 2000, pp. 599-604.

[HaA00] H.S. Hassanein, and M. M. Al-Otaibi, "Destination-Controlled Routing with Resource Reservation," IPCCC 2000, pp. 405-411.

[Hou96] C. Hou, "Routing virtual circuits with timing requirements in virtual path based ATM networks," INFOCOM '96, Vol. 1, 1996, pp. 320-328.

[Hwa93] R-H. Hwang, "Routing in High-Speed Networks," PhD Dissertation, Department of Computer Science, University of Massachusetts, USA, May 1993.

[HaZ00] F. Hao, and E. Zegura, "On Scalable QoS Routing: Performance Evaluation of Topology Aggregation," INFOCOMM 2000 Proceedings, Vol. 1, 2000, pp. 147-156.

[ISI98] A. Iwata, H. Suzuki, R. Izmailow, and B. Sengupta, "QoS Aggregation Algorithms in Hierarchical ATM Networks," IEEE Proceedings of the ICC'98, 1998, pp. 243-148.

[KhH03] Z. Khan and H. Hassanein, "QoS-Constrained IP Routing," Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering, May 2003.

[KiN00] L. King-Shan, and K. Nahrstedt, "Topology aggregation and routing in bandwidth-delay sensitive networks," IEEE Globecom 2000 Proceedings, Vol. 1, 2000, pp.410-414.

[KoK99] T. Korkmaz, and M. Krunz, "Source-Oriented Topology Aggregation with Multiple QoS Parameters in Hierarchical ATM Networks," IEEE Proceedings of the IWQoS'99, 1999, pp.137-146.

[LiM95] C. Liu and H.T. Mouftah, "Virtual call admission control-A strategy for dynamic routing over ATM network," Proceedings IEEE International Conference on communication (ICCC'95), Seattle, 1995, pp.201-205

[RoH01] A. Roumani, and H. Hassanein, "A Scheme for QoS-Based Dynamic Multicast Routing," Proceedings of Sixth IEEE Symposium on Computers and Communications 2001, pp. 132-138.

[Rou00] A. Roumani, "QoS-Based Multicast Routing," MSc. Thesis, Department of Computing and Information Science, Queen's University, Kingston, Ontario, Canada, Aug. 2000.

[ReS00] D.S. Reeves, and H.F. Salama, "A Distributed Algorithm for Delay-Constrained Unicast Routing,", IEEE/ACM Transactions on Networking, Vol. 8, No. 2, 2000, pp.239-250.

[RFC1058] C. Hedirck, "Routing Information Protocol," Internet RFC 1058, http://ds.internic.net/rfc/rfc1058.txt , June 1988.

[RFC1583] J. Moy, "OSPF Version 2," Internet RFC 1583, Mar. 1994.

[RFC2386] E. Crawley, R. Nair, B. Rajagopalan and H. Sandick, "A Framwork for QoS-based Routing in the Internet," RFC 2386, http://www.faqs.org/rfcs/rfc2386.html

[RFC2676] G. Apostolopulos, S. Kamat, R. Guerin, "QoS Routing Mechanism and OSPF Extensions," RFC 2676, Aug. 1999, 50 pages, http://www.ietf.org/rfc/rfc2676.txt

[Sal96] H.F. Salama, "Multicast Routing for Real-Time communication On High-Speed Networks", PhD Dissertation, Department of Electrical and Computing Engineering, North Carolina State University, 1996.

[ShC00] K.G. Shin and C. -C. Chou, "A Distributed Route Selection Scheme for Establishing Real-Time Channel", IEEE Transactions on Parallel and Distributed Systems, Vol. 11, No. 3, Mar. 2000, pp. 318-335.

[SGA00] V. Sarangan, D. Ghosh and R. Acharya, "Distributed QoS routing for Multimedia Traffic", Conference on Global Telecommunications, GLOBECOME '00 IEEE, Vol. 1, 2000, pp. 455-459.

[SuL98] Q. Sun and H. Langendorfer, "A new distributed routing algorithm for supporting delay-sensitive applications," Computer Communications, Vol. 21, 1998, pp.572-578.

[SMM98] R. Sriram, G. Manimaran, C. S. R. Murthy, "Preferred link delay-constrained least-cost routing in wide area networks," Computer Communications, Vol. 21, 1998, pp.1655-1699.

[SRV97] H.F. Salama, D. S. Reeves and Y. Viniotis, "A Distributed Algorithm for Delay-Constrained Unicast Routing", IEEE INFOCOM 1997, pp.84-91.

[TEL316] Data Network Design, http://www.tele.sunyit.edu/routing_principles.htm

[VPL98] S. Verma, R. Pankaj, A. Leon-Garcia, "Call admission and resource reservation for guaranteed quality of service services in internet," Computer Communications, Vol. 21, 1998, pp.362-374.

[WaC96] Z. Wang and J. Crowcroft, "Quality-of-Service Routing for Supporting Multimedia Applications," IEEE Journal on Selected Areas in Communications, Vol. 14, No. 7, 1996, pp.1228-1234.

[WaC95] Z. Wang and J. Crowcroft, "Bandwidth-delay Based Routing Algorithms," In Proceedings of the GLOBECOM '95 Conference IEEE, Vol. 3, 1995, pp. 2129-2133.

[Wax88] B.M. Waxman, "Routing of Multipoint Connection", IEEE Journal on Selected Areas in Communications, Vol. 6, No. 9, 1988, pp.1617-1622.

[Wid94] R. Widyono, "The design and evaluation of routing algorithms for real time channels," Techincal Report ICSI TR 94-024, Department of EECS, University of California at Berkeley, 1994.

[YiM02a] S. Yilmaz and I. Matta, "On the Scalability-Performance Tradeoffs in MPLS and IP Routing," Technical Report BUCS-TR-2002-013.

[YiM02b] Yilmaz, S., and I. Matta, "Unicast Routing: Cost-Performance Tradeoffs," Technical Report BUCS-TR-2002-018.

[ZKM01] B. Zhang, M. Krunz, H. Mouftah and C. Chen, "Stateless QoS routing in IP networks," In Proceedings of the IEEE GLOBECOM 2001 Conference-Global Internet Symposium, San Antonio, Texas, Nov. 2001, pp. 1600-1604

Appendix A

Confidence Intervals

Normally, confidence intervals placed on the mean values of simulation results can be used to describe the accuracy of the simulation results. Consider the results of N statistically independent simulation runs for the same experiment: $X_1, X_2, \ldots, X_N$. The sample mean, $\overline{X}$ is given as:

$$\overline{X} = \frac{\sum_{i=1}^{N} X_i}{N}$$

The variance of the distribution of the sample values, S_x^2 is:

$$S_x^2 = \frac{\sum_{i=1}^{N} (X_i - \overline{X})^2}{N-1}$$

The standard derivation of the sample mean is given by: $\frac{S_x}{\sqrt{N}}$

Under the assumption of independence and normality, the sample mean is distributed in accordance to the t-distribution, which means the sample mean of the simulation runs fall in the interval $\pm\varepsilon$ within the actual mean with a certain probability drawn from the t-distribution.

$$\varepsilon = \frac{S_x t_{\alpha/2,N-1}}{\sqrt{N}}$$

where $t_{\alpha/2,N-1}$ is the value of the t-distribution with N-1 degrees of freedom with probability $\alpha/2$. The upper and lower limits of the confidence interval regarding the simulation results are:

$$\text{Lower Limit} = \overline{X} - \frac{S_x t_{\alpha/2,N-1}}{\sqrt{N}}$$

$$\text{Upper Limit} = \overline{X} + \frac{S_x t_{\alpha/2,N-1}}{\sqrt{N}}$$