Quality of Service Routing Algorithms for
Bandwidth-Delay Constrained Applications

By

Yang Yi

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Master of Philosophy
in Computer Science Department

July 2001, Hong Kong

Copyright © by Yang Yi 2001
Authorization

I hereby declare that I am the sole author of the thesis.

I authorize the Hong Kong University of Science and Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the Hong Kong University of Science and Technology to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Yang Yi
Acknowledgments

Many thanks are given to my supervisors, Professor Samuel T. Chanson and Professor Jogesh K. Muppala, for their continuous support and help on all aspects from technical to material. Their guidance and advice helped me overcome many technical obstacles which otherwise would take much more efforts.

Thanks also go to my committee members, Professor Gary Chan and Professor Brahim BenSaoou.

The thesis is dedicated to my father and mother. It is their love, encouragement, belief and understanding that made everything I have possible.
Quality of Service Routing Algorithms for Bandwidth-Delay Constrained Applications

By

Yang Yi

This is to certify that I have examined the above MPhil thesis and have found that it is complete and satisfactory in all respects, and that any and all revisions required by the thesis examination committee have been made.

Prof. SAMUEL T. CHANSON
Supervisor

Dr. JOGESH K. MUPPALA
Supervisor

Prof. DERICK WOOD
Acting Head of Department

Computer Science Department
6 August 2001
Table of Contents

Title Page .................................................................................................................. i
Authorization Page ..................................................................................................... ii
Acknowledgements .................................................................................................... iii
Signature Page ........................................................................................................... iv
Table of Contents ...................................................................................................... v
List of Figures ........................................................................................................... vii
List of Tables ............................................................................................................. ix
Abstract ..................................................................................................................... x

Chapter

1 Introduction ............................................................................................................. 1
  1.1 Motivation .......................................................................................................... 1
  1.2 Research Goal ................................................................................................... 3
  1.3 Outline of the Thesis ........................................................................................ 3

2 QoS Routing Preliminaries .................................................................................... 5
  2.1 QoS Metrics and Constraints .......................................................................... 6
  2.2 Maintenance of State Information .................................................................. 8
  2.3 Unicast Routing Problems .............................................................................. 9
    2.3.1 On-line and Off-line Routing .................................................................. 10
    2.3.2 Constrained Routing Problems ............................................................... 10
    2.3.3 NP-Complete Theorems ........................................................................ 12
  2.4 Routing Strategies .......................................................................................... 14
    2.4.1 Centralized Model – Server-based Routing ............................................. 15
    2.4.2 Distributed Model .................................................................................... 16
      2.4.2.1 Source Routing ................................................................................ 16
      2.4.2.2 Distributed Routing ........................................................................ 17
      2.4.2.3 Hierarchical Routing ...................................................................... 18
List of Figures

2-1 Constrained Routing Problems .................................................. 12
2-2 Unicast Routing Problems .......................................................... 14
5-1 Network Topologies ................................................................... 40
5-2 Performance Results for Topology 1 ............................................. 43
5-3 Performance Results for Topology 2 ............................................. 44
5-4 Percentage of Calls as Function of Propagation Delay for 4000 Calls ..... 48
6-1 Illustrative Example ................................................................... 53
6-2 Network Topologies ................................................................... 59
6-3 Call Blocking ratio as function of the number of requests, MIRA topology with specified ingress-egress nodes, $B \in [1...5]$, $D \in [100...115\text{ms}]$ .................. 66
6-4 Call Blocking ratio as function of the number of requests, MIRA topology with specified ingress-egress nodes, $B \in [1...5]$, $D \in [100...115\text{ms}]$ .................. 67
6-5 Call Blocking ratio as function of the number of requests, MIRA topology with specified ingress-egress nodes, $B \in [1...5]$, $D \in [100...115\text{ms}]$ .................. 68
6-6 Call Blocking ratio as function of the number of requests, Expanded ANSNET topology with specified ingress-egress nodes, $B \in [1...5]$, $D \in [150...165\text{ms}]$ .................................................. 69
6-7 Call Blocking ratio as function of the number of requests, Expanded ANSNET topology with specified ingress-egress nodes, $B \in [1...5]$, $D \in [150...165\text{ms}]$ .................................................. 70
6-8 Call Blocking ratio as function of the number of requests, MIRA topology with non-specified ingress-egress nodes, $B \in [1...5]$, $D \in [100...115\text{ms}]$ .................................................. 71
6-9 Call Blocking ratio as function of the number of requests, Expanded ANSNET topology with non-specified ingress-egress nodes, $B \in [1...5]$, $D \in [125...140\text{ms}]$
List of Tables

4-1 Unicast Routing Algorithms ................................................. 36
Quality of Service Routing Algorithms for Bandwidth-Delay Constrained Applications

By

Yang Yi

Computer Science Department
The Hong Kong University of Science and Technology

Abstract

The advent of technologies like Multi-Protocol Label Switching (MPLS) and Differentiated Services (DiffServ) has enabled traffic engineering (TE) within the Internet to support multimedia applications like Voice over IP (VoIP). Although much work has been done on laying MPLS paths to optimize performance, most has focused on satisfying bandwidth requirements. Relatively little research has been done on laying paths with both bandwidth and delay constraints. In this thesis, we classify and discuss existing routing algorithms. We study the performance of several bandwidth-constrained algorithms in supporting VoIP trunks. We then present bandwidth-delay constrained routing algorithms that use knowledge of the ingress-egress node pairs in the network to reduce the rejection rates for setting up new paths. Simulation is used to evaluate the algorithms and compare their performance against existing algorithms for bandwidth constraints that have been modified to handle delay constraints. The results show that the proposed algorithms outperform all others under a wide range of workload, topology and system parameters.
Chapter 1

Introduction

1.1 Motivation

The Internet is rapidly evolving into a critical communications infrastructure with concomitant economic and commercial applications. Consequently there are growing expectations on the services provided to the users. Several recent advances in mechanisms for supporting quality of service (QoS) within the Internet, including Differentiated Services (Diffserv) [9][10], and Multi-Protocol Label Switching (MPLS) [6][29] have given hope for transforming the best-effort based Internet [39] into an infrastructure capable of supporting the QoS requirements of delay sensitive applications.

Traffic engineering (TE) has always been an integral part of managing telecommunications infrastructures [5]. Internet traffic engineering [7] is now rapidly gaining momentum as a means of assuring service quality for the users. The availability of technologies like MPLS [6] provides the necessary mechanisms to enable TE in the Internet. This gives the network service providers better control on capacity and traffic management within their networking domains. There has also been a growing movement among telecommunications and network service providers
towards a multi-service common communications infrastructure which can support both voice/video and data services. Voice over IP (VoIP) is the first application to propel this shift. There are still questions on how QoS support for high-quality VoIP services can be provided [35]. The availability of MPLS enables VoIP oriented TE within the Internet. From the perspective of this thesis, VoIP is representative of a class of applications that require the support for setting up bandwidth and delay constrained paths through the network. This thesis focus on the routing problem of MPLS label switched paths (LSPs) with bandwidth and delay guarantees for VoIP trunks in the Internet.

To support multimedia applications like VoIP that require service level assurances such as bandwidth, delay, delay-jitter and reliability, routing protocols that are capable of accommodating such constraints must be developed for IP networks. Routing with QoS constraints, known as QoS routing, is a longstanding research topic.

In traditional data networks, routing primarily focuses on connectivity. Routing protocols usually characterize the network with a single metric such as hop-count or delay and build routing tables with the only objective of minimizing the administrative cost of each path. In order to support a wide range of QoS requirements, routing protocols need to take into consideration multiple metrics. A core component of QoS routing is the identification of a path based on its QoS requirements and the resource availability of the network. In addition to increasing chances of finding a path that meets the QoS constraints, QoS routing may also improve the overall network efficiency. However these gains have to be carefully weighed against the increased cost of incurring potentially higher communication, processing, and storage overheads. As current routing protocols are already reaching the limit of feasible complexity, it is
important that the complexity introduced by the QoS support should not impair the scalability of routing protocols.

1.2 Research Goal

Most existing bandwidth-delay-constrained routing algorithms [12][14][33][45] assume the only dynamic information available is the link residual bandwidth and delay, which is maintained by protocols such as those reported in [1][22][27]. The availability of quasi-static information such as the locations of the ingress-egress nodes in the network was first exploited in [28] to provide better control on the capacity and traffic management within a network domain. While the algorithm proposed uses such information to reduce the number of request rejections, it is focused on routing in the context of setting up bandwidth guaranteed LSPs. Little research has been done on path selection algorithms using knowledge of ingress-egress nodes and considering both bandwidth and delay criteria.

The research goals of this thesis are listed in the following.

1. Evaluate the performance of bandwidth-constrained algorithms to support VoIP traffic at the VoIP trunk level. This aims at getting an insight into the issues that arise in voice traffic engineering.

2. Provide solutions to bandwidth-delay constrained routing problem taking into account knowledge of the ingress-egress node pairs.

1.3 Outline of the Thesis

The rest of the thesis is organized as follows. Chapter 2 provides a background on QoS routing. Terminology and basic concepts are presented. Different routing problems are
defined. The strengths and weaknesses of different routing strategies are discussed and compared. Chapter 3 details the VoIP traffic engineering requirements. Chapter 4 presents a survey on QoS routing algorithms for unicast applications. Basic algorithms in each problem class are reviewed and compared. In Chapter 5, the performance of several routing algorithms with bandwidth guarantees is evaluated by simulation. We concentrate on the context of routing VoIP traffic trunks in an MPLS-enabled domain. In Chapter 6, the problem of bandwidth-delay-constrained routing with available information of ingress-egress pairs is studied. New algorithms are proposed. Simulation experiments are conducted to evaluate the algorithms. Chapter 7 provides an outlook on future research in this area and concludes the paper.
Chapter 2

QoS Routing Preliminaries

QoS routing is defined as a routing mechanism in which paths are determined based on some knowledge of the resource availability in the network as well as the QoS requirements of connections [19]. The general QoS routing problem considers three issues: (1) selection of relevant QoS measures (2) determination of QoS routing protocol and (3) selection of routing algorithm.

QoS routing is different from the traditional best-effort routing. The differences manifest in three ways:

(1) QoS routing is normally connection-oriented. Best-effort routing can be either connection-oriented or connectionless.

(2) QoS routing provides the guaranteed service with resource reservation. Each feasible path identified meets the QoS requirement of each individual connection. Best-effort routing provides the best-effort service with resource sharing. Paths exhibit dynamic performance subject to the current availability of shared resources.

(3) QoS routing aims at reducing the call-blocking ratio to achieve global efficiency in resource utilization. Connections are routed in a way that the utilization of network resources is optimized. Best-effort routing considers the goals of
fairness, overall throughput and responsiveness.

In the following sections, we introduce some preliminary concepts in QoS routing. First we define the types of QoS metrics and constraints. Then we discuss different approaches to maintain network state information. Next we classify the routing problems and compare different routing strategies. Finally we describe the performance measurements.

2.1 QoS Metrics and Constraints

A network is usually modeled as a directed graph $G(V, E)$, where $V$ is the set of nodes representing switches, routers and hosts, and $E$ is the set of directed edges representing physical communication links. Every node and link has a state measured by the QoS metrics of concern. QoS metrics are the representation of a network in routing. Generally, metrics can be divided into three broad types.

**Definition 1** Let $m(i, j)$ be a QoS metric for link $l(i, j)$. For any path $P=(s, i, j, ..., l, t)$, we say that metric $m$ is

- **concave**, if $m(P) = \min[m(s, i), m(i, j), ..., m(l, t)]$. Examples are bandwidth and buffer space, etc;
- **additive**, if $m(P) = m(s, i) + m(i, j) + ... + m(l, t)$. Examples are delay, delay jitter and cost, etc;
- **multiplicative**, if $m(P) = m(s, i) \times m(i, j) \times ... \times m(l, t)$. Examples are loss probability and reliability, etc.

There are some considerations in selecting metrics to support user QoS requirements [45]:

6
(1) The metrics must reflect the basic network characteristics of interest. Since any QoS requirement is given as a set of constraints which are mapped into a combination of QoS metrics, the metrics, to some extent, determine the types of QoS requirements the network can support.

(2) The metrics should be orthogonal to each other to prune redundant information. Otherwise, interdependency introduced by redundancy makes it impossible to evaluate each metric independently and thus complicates path computation.

(3) Efficient algorithms must exist for path computation based on a single metric or a combination of metrics, so that the metrics are practical and scalable to large networks such as the Internet.

(4) Once selected, the metrics should be represented in a uniform way across independent domains to derive path metrics consistently.

The QoS requirement of a connection is given as a set of constraints, which can be link constraints, path constraints or tree constraints. A link constraint specifies the restrictions on the use of links, and is typically mapped into a concave metric. For example, a bandwidth constraint $B$ of a connection requires that each link consisting of the path must have available free bandwidth no less than $B$. A path constraint specifies the end-to-end QoS requirement on a path, which is typically mapped into an additive or multiplicative metric. For example, a delay constraint $D$ of a connection requires that the end-to-end delay of the path is no larger than $D$. A tree constraint specifies the QoS requirement for the whole multicast tree instead of a single path. An example is that a delay constraint $D$ of a multicast connection requires that the end-to-end delay from the sender to any destination is bounded by $D$. A feasible path (tree) refers to one that has enough residual (unused) resources to satisfy the QoS requirement of a connection.
2.2 Maintenance of State Information

To compute a feasible path for a new connection request, a consistent view of the network topology should be dynamically supplied to the entity responsible for path selection. State information specifies resource availability at network nodes and links, and thus has to be collected and kept up-to-date. How well state information is maintained directly affects the performance of the routing algorithm. However, state information changes dynamically due to transient load fluctuation, establishment and termination of connections, and links going up and down. It becomes increasingly difficult to gather up-to-date state information with growing network size. Therefore, a tradeoff exists between the precision of state information maintained and the overhead involved in updating the changes. There are several ways to collect, store and update state information.

- **Local state**: Each node in the network is assumed to maintain its up-to-date local state, including the residual bandwidth of the outgoing links, the queuing and propagation delay, and the availability of other resources.

- **Global state**: The global state is a consistent view of the network topology and consists of the local states at all nodes. A *link-state protocol* [36] or a *distance-vector protocol* [25][31] is used to exchange the local states among the nodes, either periodically or when triggered by certain threshold values or topology changes [1]. Link-state protocols broadcast the local state of every node to all other nodes. Distance-vector protocols exchange distance vectors among adjacent nodes. Due to the overhead concerned and non-negligible propagation delay of state messages, the global state maintained at a node is inherently imprecise. The imprecision increases as the network size grows.

- **Aggregated (partial) global state**: It is infeasible to maintain complete global
state information in large networks due to the volume of messages and communication overhead. To solve this scalability problem, a common approach is to reduce the size of the global state by aggregating information according to the hierarchical structure of the network. Each node maintains an aggregated network image [13] including different portions of the network in different level of details.

Maintenance of state information addresses a major concern of the scalability problem in QoS routing. QoS routing requires the frequent updating of dynamic network state information. The update messages consume significant network bandwidth and processing power to generate, deliver and process. Decisions of the quantity and frequency of QoS updates involve various trade-offs between performance and cost [3]. One approach to reduce the overhead is to reduce both the number and the size of the routing messages. Aggregated global state is the reduced state information obtained by such techniques as topology aggregation [24]. Another approach is to make the update interval as large as possible without compromising routing performance. This can be done by using appropriate update trigger mechanisms [4].

2.3 Unicast Routing Problems

The routing problem can be divided into two major classes: unicast routing and multicast routing. We shall focus on unicast routing in this thesis. Unicast routing can be formulated into two types of problems: on-line routing and off-line routing. Either formulation can be further classified into subclasses based on the QoS constraints.

2.3.1 On-line and Off-line Routing
On-line routing refers to the dynamic routing of connections where requests come one at a time. Off-line routing involves allocating a set of demands known in advance.

**Definition 2** The on-line routing problem is defined as follows: given a source node $s$, a destination node $t$, a set of QoS constraints $C$, and possibly an optimization goal, find the best feasible path from $s$ to $t$ which satisfies $C$.

**Definition 3** The off-line routing problem is defined as follows: given a set of communication demands $D$, each element in $D$ is specified by a source node $s_i$, a destination node $t_i$, and a set of QoS constraints $C_i$. Find one feasible path for each demand so that the constraints of all demands are simultaneously satisfied.

Under the assumption that no demand can be split, the off-line routing problem is NP-complete [20].

### 2.3.2 Constrained Routing Problems

We know from Definition 1 that for concave QoS metrics such as bandwidth and buffer space, the state of a path is determined by the state of the bottleneck link. Two types of routing problem can be defined for these metrics. One is called *link-optimization routing*. An example is widest available path routing, which is to find a path with the largest bandwidth. A slightly modified Dijkstra's algorithm or Bellman-Ford algorithm can be used to solve the link-optimization routing problem.

The other problem is called *link-constrained routing*. An example is bandwidth-constrained routing, which is to find a path whose bottleneck bandwidth is above a required value. Link-constrained routing problem can be easily reduced to the link-optimization problem.
Path-optimization routing and path-constrained routing are defined for additive and multiplicative metrics such as delay, delay jitter, cost and loss probability. For these metrics, the state of a path is determined by the combination of state over all links on the path. An example for path-optimization routing is least-cost routing, which is to find the path with the minimum total cost. Delay-constrained routing is an example of path-constrained routing, which is to find a path whose end-to-end delay is below a required value. Both path-optimization and path-constrained routing problems can be solved by Dijkstra’s or the Bellman-Ford algorithm.

The composite routing problems, which are listed in Figure 2-1, can be derived from the above four basic routing problems. For example, bandwidth-constrained least-cost routing problem belongs to the link-constrained path-optimization routing (LCPO), and can be solved by Dijkstra’s or the Bellman-Ford algorithm. Other problem classes that are solvable in polynomial time by a modified shortest path algorithm include link-constrained link-optimization routing (LCLO), multi-link-constrained routing (MLC), link-constrained path-constrained routing (LCPC), and path-constrained link-optimization routing (PCLO).

Routing problems of particular interest are path-constrained path-optimization routing (PCPO) and multi-path-constrained routing (MPC), both of which are NP-complete problems. An example of PCPO is delay-constrained least-cost routing, which is to find a path with minimum cost that can satisfy the delay constraint. Delay-delay-jitter-constrained routing is an example of MPC. It is to find a path that satisfies both the delay and delay-jitter constraints. It is proven that a routing problem is NP-complete if the following conditions are satisfied [45]: (1) the metrics are independent (2) more than one metric is additive or multiplicative and (3) more than
one metric takes real values or unbounded integer values.

Figure 2-1: Constrained Routing Problems

2.3.3 NP-Complete Theorems

As discussed in the above section, finding a path subject to multiple constraints is inherently hard. The computation complexity is primarily determined by the combination of the metrics. We here list three general NP-completeness theorems for additive and multiplicative metrics. The proofs for them can be found in [45].

**Theorem 1** The $n$ Additive Metrics Problem: given a network $G = (V, E)$, $n$ additive metrics $d_1(e), d_2(e), ..., d_n(e)$ for each $e \in E$, two specified nodes $i, m$, and $n$ positive integers $D_1, D_2, ..., D_n$, $(n \geq 2, d_i(e) \geq 0, D_i \geq 0$ for $i = 1, 2, ..., n)$, the problem of deciding if there is a simple path $p = (i, j, k, ..., l, m)$ which satisfies the following constraints $d_i(p) \leq D_i$, where $i = 1, 2, ..., n$ is NP-complete.
Theorem 2 The \( n \) Multiplicative Metrics Problem: given a network \( G = (V, E) \), \( n \) multiplicative metrics \( d_1(e), d_2(e), \ldots, d_n(e) \) for each \( e \in E \), two specified nodes \( i, m \), and \( n \) positive integers \( D_1, D_2, \ldots, D_n \), \( n \geq 2, d_i(e) \geq 1, D_i \geq 1 \) for \( i = 1, 2, \ldots, n \), the problem of deciding if there is a simple path \( p = (i, j, k, \ldots, l, m) \) which satisfies the following constraints \( d_i(p) \leq D_i \), where \( i = 1, 2, \ldots, n \) is NP-complete.

Theorem 3 The \( n \) Additive and \( k \) Multiplicative Metrics Problem: given a network \( G = (V, E) \), \( n \) additive and \( k \) multiplicative metrics \( d_1(e), d_2(e), \ldots, d_{n+k}(e) \) for each \( e \in E \), two specified nodes \( i, m \), and \( n+k \) positive integers \( D_1, D_2, \ldots, D_{n+k} \), \( n \geq 1, k \geq 1, d_i(e) \geq 1, D_i \geq 0 \) for \( i = 1, 2, \ldots, n \), \( D_i \geq 1 \) for \( i = n+1, n+2, \ldots, n+k \), the problem of deciding if there is a simple path \( p = (i, j, k, \ldots, l, m) \) which satisfies the following constraints \( d_i(p) \leq D_i \), where \( i = 1, 2, \ldots, n+k \) is NP-complete.

The three Theorems above show that the problem of finding a path subject to constraints on two or more additive and multiplicative metrics in any possible combination is NP-complete. This indicates that the complexity of a unicast routing problem is determined, to some extent, by the number of additive and multiplicative metrics it addresses. According to this criterion, we classify unicast routing problems into three classes shown in Figure 2-2.

- **Non-Path-Constrained Problems (NPCP)**: Both link-constrained and link-optimization problems, and the combinations of them belong to NPCP.
- **Single-Path-Constrained Problems (SPCP)**: Path-constrained and path-optimization problems are in this class. Moreover, link constraints can be added to both of them to form SPCP.
- **Multiple-Path-Constrained Problems (MPCP)**: MPCP consists of
combinations of SPCP problems.

Formal definitions of these three classes of unicast routing problems are given in Chapter 4, which are followed by discussion and comparison of basic routing algorithms in each class.

Figure 2-2: Unicast Routing Problems

2.4 Routing Strategies

Implementation of routing protocol falls into two models: centralized and distributed. In a centralized model, a single entity maintains a complete state information database and is responsible for path selection. While in a distributed model, all nodes are required to keep their own network topology databases. Based on how the state information is maintained and how the search of the feasible path is carried out, we classify the QoS routing strategies into four categories: server-based routing, source routing, distributed routing, and hierarchical routing. Server-based routing follows a centralized model while the other three follow a distributed model.
2.4.1 Centralized Model – Server-based Routing

In server-based routing, a single entity in the network, called route server [2] or network manager [34], keeps the complete network topology database and is responsible for finding a feasible path for every connection setup request with QoS constraints. A request either directly reaches the route server or first arrives at a source node that forwards the request to the route server. The route server generates an explicit route and sends it back to the source node. The source node then uses a signaling protocol such as RSVP or LSP to set up the path to the destination and reserve resources on each link along the path.

Server-based routing has several advantages. First, the policy capable admission control architecture proposed by the Internet Engineering Task Force (IETF) is structured around a client-server model. Policy servers are responsible for making policy decisions for incoming requests. Since part of the processing of a request is to contact a policy server, this facilitates accessing information in a remote route server. Thus, potential access to a route server can be easily piggybacked on the policy server access. Second, performing most QoS-capable tasks at the server allows other nodes in the network to be kept simple, and thus much more limits the extent of their QoS awareness. Therefore, costly changes in the network infrastructure caused by introduction of QoS capabilities can be minimized.

A big concern with server-based routing is the scalability problem. Due to the use of a centralized route server, performance and reliability also suffers. Since every request has to be approved by the route server, when the number of requests is high, the server may be the performance bottleneck. Techniques like path caching have been proposed to address this problem.
2.4.2 Distributed Model

2.4.2.1 Source Routing

In source routing every node in the network maintains its own state information database. A feasible path is locally computed at the source node independent of all other nodes. A control message is then sent along the path to inform the intermediate nodes of their precedent and successive nodes. A link-state protocol is used to update the global state at every node.

The major advantage of source routing is its simplicity. By transforming a distributed problem into a centralized one, it avoids problems such as distributed state snapshot, deadlock detection, and distributed termination. Also, loop-free routes can be guaranteed. Moreover, it is much easier to design centralized heuristics for some NP-complete routing problems.

However, since all the computation work is done at the source, the overhead can be excessively high especially when dealing with multiple constraints. Moreover, since dynamic network parameters such as bandwidth and delay change very rapidly, to keep link state information up-to-date, the global state maintained at each node has to be updated frequently, causing high communication overhead for large-scale networks. In other words, source routing has a scalability problem.

2.4.2.2 Distributed Routing

Distributed routing refers to the situation where the routing is done on a hop-by-hop
basis. A feasible path can be pre-computed or searched on demand for a setup request. In the case of pre-computing paths, each node in the network calculates in advance forwarding entries for every possible destination. A forwarding entry stores the next hop on the “best” path to a destination and this is updated periodically. For instance, each node keeps an entry of the next hop on the least-delay path to each destination. With the pre-computed forwarding entries, the routing process is reduced to looking up the routing tables. In the on-demand case, the searching of a feasible path is carried out on arrival of a setup request. The path is computed by a distributed computation. Intermediate nodes between the source and the destination participate in the path computation and exchange control messages. The state information kept at each node is selectively used for the path search. Most distributed routing algorithms need a distance vector protocol or a link state protocol to maintain a global state at every node.

Distributed path computation shortens the routing response time and makes the algorithm more scalable. Selective use of the state information maintained at each node increases the use of up-to-date local state, and thus makes the search result more accurate. In some algorithms, the routing decision and optimization are done based entirely on the local states [14] [37]. Searching multiple paths in parallel for a feasible path is made possible, thereby increasing the chance of success.

Most distributed algorithms require the global state and share more or less the same problems of source routing algorithms. Control messages exchanged among nodes take up network resources and can cause a communication burden. Loops may occur due to inconsistency in the global state at different nodes. This would generally cause routing to fail because the distance vectors do not provide sufficient information for an
alternative path. Moreover, it is very hard to design efficient distributed heuristics for NP-complete routing problems.

2.4.2.3 Hierarchical Routing

In hierarchical routing, physical nodes of the network are divided into groups, which are further clustered into higher-level groups recursively, creating a multilevel hierarchy. Each group is aggregated into a simpler topology. This step is called *topology aggregation*. Aggregate state information is maintained at each physical node, which contains detailed state information about the nodes in the same group and aggregate state information about the other groups. Source routing is performed at the source node to find a feasible path, which may contain logical nodes representing groups and logical links clustered by physical links. A control message or packet is then sent to travel along this path establishing the connection. When a border node of a group represented by a logical node receives the message, it uses source routing to expand the path through the group.

A big advantage of hierarchical routing is that it is able to cope with the scalability problem in large networks. The reason why hierarchical routing scales well lies in the fact that each node only keeps a partial global state with the size being logarithmic to the size of the complete global state. Well-studied source routing algorithms can be directly used at each hierarchical level to find feasible paths based on the aggregated state. Therefore, hierarchical routing retains many advantages of source routing. Since many nodes share the routing computation, it also has some advantages of distributed routing.

However, how to aggregate the global state information effectively and accurately is
still an open problem. First, aggregation algorithms tend to suffer from distortion, which means the resulted aggregated topology deviates from the original one [32]. Second, additional imprecision is introduced due to the aggregated network state. A simpler topology or even a logical node is used to represent a large subnet with complicated internal structure. Hence, it is hard to estimate actual available resources due to the hidden internal structure. Finally, besides the need to consider the amount of resources that are available, it also needs to consider the level of certainty with which these resources are indeed available [21].

2.5 Performance Measures

There exist many performance measures to evaluate a routing scheme, such as computation complexity, communication overhead and cost, etc. In particular, we may also evaluate them according to the following abstract criterion.

- **Efficiency**: Most source routing algorithms for the NP-complete problems are not scalable due to prohibitively high time complexity, especially for multicast routing. Efficient algorithms are required to achieve a good balance between the computation time and connection-success ratio.

- **Simplicity**: The simplicity of a routing algorithm in terms of time/logical complexity often allows efficient implementation, debugging and evaluation.

- **Generality**: Since multimedia applications tend to have diverse QoS requirements on bandwidth, delay, delay jitter, cost, and so on, it would be beneficial to develop a generic routing algorithm for any constraints instead of implementing different routing algorithms for different types of QoS requirements independently.

- **Extensibility**: As the network infrastructure evolves and capacity increases, new
applications are made possible. It requires the routing algorithms to be extendable to accommodate new service types. It is important to design extensible algorithms and make them adapt to new applications, because as the networks become increasingly complex, the deployment of new routing algorithms is very costly and problem-prone.

2.6 Summary

An overview of QoS routing is given in this chapter. QoS routing is a key network function. It has two objectives: finding routes that satisfy the QoS constraints and making efficient use of the network resources. The goal of unicast routing is to find a constrained network path between two end nodes. Based on the way the state information is maintained, the routing strategies can be divided into four classes: (1) server-based routing, (2) source routing, (3) distributed routing and (4) hierarchical routing. Many routing problems are NP-complete. According to the complexity of the routing problem they address, the existing unicast routing algorithms can be classified into three classes: non-path-constrained, single-path-constrained and multiple-path-constrained routing. These concepts are useful in understanding the subsequent chapters of the thesis.
Chapter 3

Voice Traffic Engineering

In this chapter, we present the characteristics of voice traffic and the quality of service issues unique to packet networks. Then we make a brief introduction to MPLS and discuss traffic engineering oriented QoS routing at the end.

3.1 Voice Traffic Characteristics

The steps involved in delivering VoIP include: sampling, digitization & encoding, encapsulation, transmission, decoding, buffering and play-out. ITU-T standard voice codec algorithms include [44]: G.711 at 64 Kbps using PCM, G.726 using ADPCM at 40, 32, 24, and 16 Kbps. CELP is used for G.728 at 16 Kbps, G.729/G.729A at 8 Kbps. G.723.1, based on MP-MLQ technology with two transmission rates i.e., 5.3 and 6.3 Kbps, generally provides good speech quality. Details about these codecs can be found in [44].

VoIP QoS issues unique to packet networks are delay, jitter and loss. Delay gives rise to echo and talker overlap problems. Echo is a significant quality problem when round-trip delays are greater than 50 ms, while the talker overlap problem becomes significant when the one-way delay is greater than 400 ms. Since packets experience varying delays in the network, inter-packet time on the receiver side is not constant.
even if it is so on the sender’s side. A play-out buffer is employed to filter out the jitter, which introduces further delay. A late or lost packet will affect the quality of the voice received. To recover from loss packets, voice may be interpolated from the last received packet to fill the place of the lost packet.

3.2 Multiprotocol Label Switching

MPLS [46] is a fast forwarding scheme based on labels that provides a link layer independent transport framework for IP. Each MPLS packet contains a header with 20-bit label, a 3-bit experimental field, a 1-bit stack indicator, and an 8-bit time-to-live field. In an ATM environment, the label information is carried in the virtual circuit/virtual path identifier (VPI/VCI). An MPLS-capable router is called a label-switching router (LSR). The MPLS router examines the label (and possibly the other fields of the MPLS header) of the incoming packet in making the forwarding decisions. The MPLS headers are attached to the IP packet at the ingress router to an MPLS capable network. At each LSR, the label field is used to look up a forwarding table. The incoming label field is then replaced by an outgoing label before the packet is forwarded to the next LSR along the path. MPLS sets up explicit paths, called Label-switched paths (LSPs) between each ingress LSR and egress LSR within the MPLS domain. The LSP setup is accomplished using signaling protocols such as the Resource Reservation Protocol (RSVP) or Label Distribution Protocol (LDP). The setting up of LSPs within the domain, satisfying certain QoS attributes, is the focus within this thesis.

3.3 QoS Routing for Traffic Engineering

Voice traffic engineering can be broken down into three distinct layers: (1) off-line
voice traffic engineering, (2) connection admission and/or connection routing, (3) dynamic traffic management [26]. The operation of QoS routing, when aiming primarily at traffic engineering, is characterized by a long timescale (long-term traffic variation) and a coarse granularity of the traffic flows it handles (traffic aggregates) [3]. In such an environment, the goal of QoS routing is maximization of network performances in the presence of slowly changing traffic patterns. This is achieved through continuous measurements of traffic patterns and the computation of paths on which to route traffic aggregates to optimize various performance measures.

3.4 Summary

VoIP applications require support for setting up bandwidth-delay-constrained paths in the network. QoS issues unique to VoIP in packet networks are delay, jitter and loss. MPLS adds connection-oriented capabilities to the connectionless IP-architecture. We can use MPLS as the enabling technology for VoIP oriented TE within the Internet. In this context, the granularity of routing decision is VoIP trunk instead of individual flows.
Chapter 4

Related Work

As discussed in Chapter 2, unicast routing problems can be divided into three classes: non-path-constrained, single-path-constrained, and multiple-path-constrained. Both non-path-constrained and single-path-constrained problems are solvable in polynomial time. Multiple-path-constrained problems, which involve two or more additive and multiplicative constraints, are NP-complete. In this chapter, we discuss and compare algorithms proposed in each class. See Table 4-1 for a summary of the comparison.

4.1 Non-Path-Constrained Routing Problems and Algorithms

Definition 4 Non-Path-Constrained Problem (NPCP): Given a network \( G = (V, E) \), each link \( l(i, j) \in E \) is associated with \( K \) concave QoS metrics \( m_k(i, j) \in \mathbb{R}^+ \), \( k = 1, 2, \ldots, K \). Given \( K \) constraints \( \{ M_k, \; k = 1, 2, \ldots, K \} \), the problem is to find a path \( P \) from a source node \( s \) to a destination node \( t \) such that:

\[
m_k(P) \equiv \min_{i(j, j) \in P} m_k(i, j) \geq M_k, \; k = 1, 2, \ldots, K.
\]

Bandwidth-constrained routing is the most commonly addressed NPCP problem. We discuss in the following the algorithms developed for it.
WAPF [35] – The Widest Available Path First algorithm (WAPF) chooses the path with the maximum available capacity from the source to the destination. It explores a set of alternate paths and attempts to load-balance the network traffic.

To provide more flexibility other than choosing the widest path, a technique called topology filtering is proposed to deal with concave metrics. All nodes and links without sufficient resources to satisfy the required constraints are first eliminated from the graph before computing a feasible path. Therefore, any path obtained from the remaining graph satisfies the requirements.

Min-Hop [22] – The Minimum Hop algorithm (Min-Hop) adopts topology filtering and selects the path with the least number of feasible links computed by Dijkstra’s algorithm. The justification is that the few number of hops the path contains, the less resource it consumes. Thus the network has more remaining resources to accommodate future requests.

WSP [22] – The Widest-Shortest Path algorithm (WSP) finds a feasible minimum-hop path between the source and destination such that the chosen minimum-hop path has the maximum residual bandwidth.

Recently published works take into consideration the impact of current routing on future requests to improve the overall network utilization [20][28].

MIRA [28] – The Minimum Interference Routing algorithm (MIRA) exploits the knowledge of ingress-egress pairs in finding a feasible path. The idea is that a newly routed connection should follow a path that does not “interfere too much” with a path that may be critical to satisfy a future demand. Based on the ingress-egress
information, critical links are identified as those links for which if a path is routed through it, the maxflow value of one or more ingress-egress pairs decreases. Weights are then assigned to links according to their “criticality”. The algorithm aims at avoiding the critical links while making path selection.

BI [20] – The Blocking Island algorithm (BI) addresses the problem of bandwidth-constrained routing in an off-line manner. The problem is defined as Resource Allocation in Networks (RAIN) in which a set of demands is known in advance and the goal is to find one feasible route for each demand. The paper introduced an abstraction technique from Artificial Intelligence called Blocking Island (BI). A $\beta$-blocking island ($\beta$-BI) for a node $x$ is defined as the set of all nodes of the network that can be reached from $x$ using links with at least $\beta$ available resources, including $x$. With possible $\beta$ values in decreasing order, the network topology is built into a Blocking Island Hierarchy (BIH). The algorithm attempts to preserve bandwidth connectivity so that the risk of future allocation failure is reduced. A demand is routed on the path which causes the fewest splitting of BIs in the BIH.

We see that all the above algorithms use server-based or source routing strategy. They characterize the network by metrics of bandwidth and hop-count. Most algorithms transform the routing problem into a shortest-path problem and then solve it by Dijkstra's or the Bellman-Ford algorithm. Since the problem is simple, efforts are put on optimizing the network efficiency.

4.2 Single-Path-Constrained Routing Problems and Algorithms

Definition 5 Single-Path-Constrained Problem (SPCP): Given a network $G = (V, E)$, each link $l(i, j) \in E$ is associated with an additive or multiplicative QoS
metric $d(i, j) \in R_0^+$ and $K$ concave QoS metrics $m_k(i, j) \in R_0^+$, $k = 1, 2, \ldots, K$. Given $K+1$ constraints $\{D, M_k, k = 1, 2, \ldots, K\}$, the problem is to find a path $P$ from a source node $s$ to a destination node $t$ such that:

1. $d(P) = \sum_{(i,j) \in P} d(i, j) \leq D$;

2. $m_k(P) = \min_{(i,j) \in P} m_k(i, j) \geq M_k, k = 1, 2, \ldots, K$.

The most interesting SPCP problem is bandwidth-delay-constrained routing. We present some algorithms addressing this problem in the following.

The Wang-Crowcroft Algorithm [45] – This source routing algorithm proposed by Wang and Crowcroft is based on Dijkstra's algorithm. First, topology filtering is applied to remove infeasible links with regard to bandwidth. Then a shortest path in terms of delay is found in the remaining graph. The path is feasible if and only if it satisfies the delay constraint.

SWP [45] – The Shortest Widest Path algorithm (SWP) is a distributed bandwidth-delay-constrained routing algorithm. It finds a feasible widest available path between the source and the destination in terms of bandwidth. If multiple paths exist, the path with the minimum delay, called the shortest-widest path, is selected. Both the modified Dijkstra's and Bellman-Ford algorithms can be used to compute the shortest-widest path.

The Shin-Chou Algorithm [42][43] – The algorithm proposed by Shin and Chou is a flooding-based algorithm where routing messages are flooded from the source towards the destination. Each message contains the accumulated delay of the path it traversed so far. A routing message is forwarded by an intermediate node if it is the
first such message received by the node or it carries a better delay than the previously received messages. Candidate out-going links are searched to forward the routing message. An out-going link is considered as a candidate if its delay plus the message's accumulated delay does not exceed the end-to-end delay requirement. A message arriving at the destination finds a feasible path, which is the one it has traversed.

The Lui-Nahrstedt Algorithm [32][33]

Lui and Nahrstedt proposed hierarchical routing algorithms to solve the bandwidth-delay-constrained problem in an IP network. The whole network is partitioned into disjoint domains and represented by a two-level hierarchical structure. Therefore the routing is twofold: it includes inter-domain routing and intra-domain routing. Each domain is aggregated into a simple star with inter-border links called bypasses to reduce the topology information broadcasted to other domains. A line segment in the delay-bandwidth space is used to represent a logical link instead of simple numerical values. Two line segments, called inadmissible line and admissible line, are identified which divide the whole delay-bandwidth space into three areas: admissible region, inadmissible region and uncertain region.

LSRA [32] – One routing algorithm proposed is called Line-Segment Routing Algorithm (LSRA). LSRA is based on a modified Dijkstra's algorithm, which is augmented to deal with the line segment representation. The network that each node sees is named as the node-view network (NVN). The source node performs LSRA in its NVN searching a feasible inter-domain path towards the destination. Then a message or a packet is sent from the source to travel along the inter-domain path found. A feasible intra-domain path is searched when the message or packet arrives at each intermediate domain.
Ticket-based Routing Algorithm [33] – The other algorithm is based on a distributed process called ticket-based probing. A ticket represents the permission of searching one path. Tickets can only be generated by the source node and the number of tickets at the source is determined by the region in which the (Delay, Bandwidth) point specifying the QoS constraints lies. On receiving a probe carrying one or more tickets, a node finds candidate neighbors to forward the tickets. The number of tickets sent to a candidate neighbor is determined by the constraints, the admissible and the inadmissible lines of that neighbor. If no candidate is found, all the tickets are dropped. The searching process terminates when a probe reaches the destination, which means a feasible path is found, or there is no ticket in the network.

4.3 Multiple-Path-Constrained Routing Problems and Algorithms

Definition 6 Multiple-Path-Constrained Problem (MPCP): Given a network $G = (V, E)$, each link $l(i, j) \in E$ is associated with $N \geq 2$ additive or multiplicative QoS metric $d_n(i, j) \in R^+_0$, $n = 1, 2, \ldots, N$ and $K$ concave QoS metrics $m_k(i, j) \in R^+_0$, $k = 1, 2, \ldots, K$. Given $N + K$ constraints $\{ D_n, M_k, n = 1, 2, \ldots, N \, , \, k = 1, 2, \ldots, K \}$, the problem is to find a path $P$ from a source node $s$ to a destination node $t$ such that:

1. $d_n(P) \equiv \sum_{l(i, j) \in P} d_n(i, j) \leq D_n, \, n = 1, 2, \ldots, N$;

2. $m_k(P) \equiv \min_{l(i, j) \in P} m_k(i, j) \geq M_k, \, k = 1, 2, \ldots, K$.

Past efforts have been put on proposing heuristics and approximations to the NP-complete MPCP problem. The research can be divided into two directions: source routing and distributed routing. In the following, we first discuss some source routing algorithms and then present several distributed routing algorithms.
The Limited Granularity Heuristic [47] – The limited granularity heuristic is a heuristic for the NP-complete two-additive-constrained routing problem. We know that if one of the two metrics takes bounded finite values, the problem can be solved in polynomial time. The limited granularity heuristic uses a bounded finite range that contains $X$ elements to approximate one metric. Let $m$ be the metric to be approximated and $C$ be the constraint. A mapping scheme, $r_1, r_2, ..., r_x$, is generated where $0 < r_1 < r_2 < ... < r_x = C$. Any value $w \in [0, C]$ of metric $m$ is mapped into $r_i$ if and only if $r_{i-1} < w \leq r_i$. The new problem then can be solved in polynomial time by an extended Bellman-Ford algorithm.

The Limited Path Heuristic [47] – The limited path heuristic defines optimal path by partial order. A path $P$ from source $s$ to destination $t$ with two metrics $m_1$ and $m_2$ is optimal if there does not exist a path $Q$ from $s$ to $t$ such that $m_1(Q) < m_1(P)$ and $m_2(Q) < m_2(P)$. An extended Bellman-Ford algorithm can be used to find the optimal path by storing at each node all sub-optimal paths currently found in the search process. However, the complexity is too high. The limited path heuristic reduces the complexity by restricting the number of paths maintained at each node to be at most $X$.

TAMCRA [38] – The Tunable Accuracy Multiple Constraints Routing algorithm (TAMCRA) is an approximation of the NP-complete multi-constrained routing problem. The idea is to reduce the original problem to a shortest path problem by representing the path length as a non-linear combination of multiple metrics. Let $l_k(i, j), k = 1, 2, ..., m$, be $m$ metrics associated with the link $(i, j)$. For a path $P$, the length $l(P)$ is defined as $l(P) = \left( \sum_{k=1}^{m} \left( \frac{l_k(P)}{L_k} \right)^{q/k} \right)^{\frac{1}{q}}$, which is reduced to $l(P) = \max\left( \frac{l_1(P)}{L_1}, ..., \frac{l_m(P)}{L_m} \right)$ when $q \to \infty$. With this definition of path length, a
The $k$-shortest path algorithm is used to compute the shortest path.

SSR+DCCR [23] — Search Space Reduction + Delay Cost Constrained Routing algorithm (SSR+DCCR) solves the delay-constrained minimum-cost path problem. The algorithm converts the delay-constrained least-cost problem (DCLC) into a delay-cost-constrained problem (DCC) by defining an appropriate cost bound for DCC. With the introduced cost constraint, SSR + DCCR finds a feasible path in a similar way as TAMCRA [38] except the definition of path length is different. With the objective to minimize the cost, SSR-DCCR defines the path length as

$$W(P^*_n) = \begin{cases} \frac{D(P^*_n)}{1 - C(P^*_n) / \Delta c}, & \text{if } D(P^*_n) \leq \Delta d \land C(P^*_n) \leq \Delta c \\ \infty, & \text{otherwise} \end{cases}$$

which gives priority to low-cost paths, while, in TAMCRA, all link metrics are treated equally. To further reduce search space, BG algorithm proposed in [11] is first run to find a tighter cost bound for SSR+DCCR. Then SSR+DCCR searches a feasible path by a $k$-shortest path algorithm with the path length defined above.

According to the number of paths explored simultaneously in the search process, distributed MPCP routing algorithms can be classified into two categories: single-path routing and multi-path routing. In single-path routing, one path is explored at a time to search for a feasible path, while several paths are tried in parallel in multi-path routing.

DCUR [41] — The Delay-Constrained Unicast Routing algorithm (DCUR) is a single-path routing algorithm, which solves the delay-constrained least-cost routing problem. Each node in the network maintains the following state information: the cost of each outgoing link, the delay of each outgoing link, a cost vector, a distance vector and a routing table. The cost (delay) vector at node $v$ contains, for each node $w$, an entry storing the cost (delay) value of the least cost (delay) path from $v$ to $w$ and the
next hop on the path. When a connection setup request arrives, the source node first examines its delay vector to see whether a feasible path exists, i.e. whether the least delay from the source to the destination can satisfy the delay constraint. If the answer is positive, a control message is sent to construct the routing 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 path directed by the cost vector, and the other \((i, k)\) is on the least delay path directed by the delay vector. Link \((i, j)\) has priority over link \((i, k)\), as long as adding the least-delay path from \( j \) to the destination does not violate the delay constraint.

Multi-path routing has recently been extensively studied. Some algorithms proposed separate routing and resource reservation [14], while others combine these two steps [16] [34]. Separating routing and resource reservation simplifies the protocol design. In a dynamic network, however, resource availability may change rapidly and the route information may be outdated. A path selected as feasible during the execution of the routing algorithm may lack the resources in the reservation step. Combing the two steps was suggested to overcome this problem [17][40]. However, reserving resources on multiple path simultaneously for each connection request causes the \textit{over reservation} problem. Furthermore, the resource availability characteristics can be greatly changed due to over reservation, which decrease the precision of the network state information. Early-release policy is usually adopted to offset the effects of over reservation.

The Chen-Nahrstedt Algorithm [14] – Chen and Nahrstedt proposed a routing framework which provides a common basis for distributed QoS routing regardless the specific metrics involved. The connection establishment protocol consists of two
phases: probing phase and acknowledgement phase. The probing phase does the QoS routing and the acknowledgement phase does the resource reservation. A generic distributed routing algorithm (DRA) is derived from this two-phase connection establishment protocol based on selective probing. After a connection request arrives, probes are flooded selectively along those paths which satisfy the forwarding condition. Upon receipt of probes, the destination selects a tentative path among the detected feasible paths based on the optimization information carried by the probes. Then an acknowledgement message is sent back to the source along the tentative path, and reserves resources along the way. Various concrete DRAs have been derived to support a variety of QoS constrains on bandwidth, delay, delay jitter, as well as their combinations.

Cidon et al. Algorithm [16] – The Cidon et al. algorithm refers to a family of distributed multi-path algorithms that combine resource reservation with the routing process. When a node wishes to establish a connection, it first finds a restricted sub-graph called “diroute”, which contains links leading to the destination at a “reasonable” cost. A link is eligible if it has enough resources to be potentially part of the requested route. Request messages are flooded along the eligible links in the diroute. On receiving a Request message, a node reserves the needed resources in all eligible outgoing links and forwards the Request message on all these links. A connection is established when the first Request researches the destination. The algorithms release resources from segments of the diroute as soon as they learn that these segments are inferior to another segment. The algorithms vary according to the trade-off between routing complexity and path optimality.

Parallel Probing Algorithm [34] – The Parallel Probing algorithm was proposed with
the objective to improve Average Call Acceptance Rate (ACAR), Average Call Setup Time (ACST) and Average Routing Distance (ARD). The main idea is to simultaneously probe \( k \) different paths using \( k \) different heuristics, one for each path. A subset of the nodes on the least-cost path between the source and the destination are chosen as intermediate destinations (ID) with \( ID_0 \) being the source and \( ID_n \) the destination. \( ID_0 \) initiates sending probe packets to \( ID_1 \) in parallel on \( k \) different paths by employing \( k \) different heuristics. The first probe arriving at \( ID_1 \) sets up a path segment between \( ID_0 \) and \( ID_1 \), and causes later probes to be rejected. Segment by segment, a feasible path is found by repeating the above process until a probe reaches the destination. Then the probe packet is sent back to the source to prune loops and release excess resources.

One big advantage of multi-path routing over single-path routing is its short connection setup time. When multiple reservation processes are conducted concurrently for the same connection, a reservation failure in one (or more) links does not slow down the reservation process in other links. Another advantage is that when several feasible paths are available, the best one based on certain optimization criteria can be selected. Work has been done to study the impact of resource reservation on multi-path routing [17][48]. Zhong and Yuan did simulation experiments with different work loads [48]. They found that in most cases, multi-path routing with resource reservation outperforms single-path routing in finding feasible paths. Cidon et. al. showed by analysis that multi-path routing has slightly better throughput than single-path routing when no retries are allowed, and slightly worse when one or two retries are used [17]. However, multi-path routing has significantly shorter connection establishment time than single-path routing. These results show that multi-path routing is an attractive consideration in future bursty applications such as World Wide
Web (WWW) browsing.

4.4 Summary

We here reviewed the existing unicast routing algorithms. The polynomial-complexity routing problems are well solved by the shortest path based algorithms while heuristics have been proposed for the NP-complete routing problems. Existing solutions for the NP-complete problems try to tradeoff between routing performance and computation/communication overhead. However, most algorithms do not scale well due to the need to maintain the global state. Some algorithms which only depend on local states have been proposed. Hierarchical routing with multiple constraints, although providing a scalable solution, is still an unsolved problem.
<table>
<thead>
<tr>
<th>Routing problem</th>
<th>Algorithm</th>
<th>Time complexity</th>
<th>Routing strategy</th>
<th>Communication complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Maintaining state</td>
</tr>
<tr>
<td>Non-Path-Constrained</td>
<td>WAPF [35]</td>
<td>$O(v \log v)$</td>
<td>Source</td>
<td>Global</td>
</tr>
<tr>
<td>Problem</td>
<td>Min-Hop [22]</td>
<td>$O(v \log v)$</td>
<td>Source</td>
<td>Global</td>
</tr>
<tr>
<td></td>
<td>WSP [22]</td>
<td>$O(ve)$</td>
<td>Source</td>
<td>Global</td>
</tr>
<tr>
<td></td>
<td>MIRA [28]</td>
<td>$O(kv \varepsilon^2 + \varepsilon)$</td>
<td>Server-based</td>
<td>Global</td>
</tr>
<tr>
<td></td>
<td>BI [20]</td>
<td>$O(be)$</td>
<td>Source</td>
<td>Global</td>
</tr>
<tr>
<td>Multi-Path-Constrained</td>
<td>Wang-Crowcroft [45]</td>
<td>$O(v \log v + \varepsilon)$</td>
<td>Source</td>
<td>Global</td>
</tr>
<tr>
<td>Problem</td>
<td>SWP [45]</td>
<td>$O(ve)$</td>
<td>Distributed</td>
<td>Global</td>
</tr>
<tr>
<td></td>
<td>Shin-Chou [42][43]</td>
<td>$O(e)$</td>
<td>Distributed</td>
<td>Local</td>
</tr>
<tr>
<td></td>
<td>Lui-Nahrstedt [32][33]</td>
<td>$O(v \log v)$</td>
<td>Hierarchical</td>
<td>Aggregated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$O(e)$</td>
<td>Hierarchical</td>
<td>Aggregated</td>
</tr>
<tr>
<td>Multiple-Path-Constrained</td>
<td>Limited Granularity</td>
<td>$O(xve)$</td>
<td>Source</td>
<td>Global</td>
</tr>
<tr>
<td>Problem</td>
<td>Heuristic [47]</td>
<td>$O(x^2v)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Limited Path Heuristic</td>
<td>$O(kv \log(kv) + k^3me)$</td>
<td>Source</td>
<td>Global</td>
</tr>
<tr>
<td></td>
<td>[47]</td>
<td>$O(k^3v)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TACMRCA [38]</td>
<td>$O(kv \log(kv) + k^3me)$</td>
<td>Source</td>
<td>Global</td>
</tr>
<tr>
<td></td>
<td>SSR+DCCR [23]</td>
<td>$O(kv \log(kv) + k^3me)$</td>
<td>Source</td>
<td>Global</td>
</tr>
<tr>
<td></td>
<td>DCUR [41]</td>
<td>$O(e^2)$</td>
<td>Distributed</td>
<td>Global</td>
</tr>
<tr>
<td></td>
<td>Chen-Nahrstedt [14]</td>
<td>$O(e)$</td>
<td>Distributed</td>
<td>Local</td>
</tr>
<tr>
<td></td>
<td>Cidon et. al. [16]</td>
<td>$O(e)$</td>
<td>Distributed</td>
<td>Global</td>
</tr>
<tr>
<td></td>
<td>Parallel Probing [34]</td>
<td>$O(e)$</td>
<td>Distributed</td>
<td>Global</td>
</tr>
</tbody>
</table>

Table 4-1: Unicast Routing Algorithms
Chapter 5

Performance of Bandwidth-Constrained QoS Routing Algorithms

In this chapter, we consider the traffic engineering issues arising in the routing of VoIP traffic trunks over an IP network. MPLS is used as the technology to support the routing of the VoIP trunks. We examine several bandwidth-constrained routing algorithms for routing VoIP trunks. Their performance with respect to the trunk blocking probability and the propagation delays of the paths is examined. We find that compared to shortest path based routing algorithms, those algorithms that try for alternate paths exhibit better performance.

5.1 System Model

The architectural model that we consider in this chapter is a backbone IP network with MPLS capability [35]. We consider an IP network consisting of \( n \) routers interconnected in a sparse network topology. A subset of these routers form the ingress-egress router pairs between which VoIP trunks are to be setup. The VoIP trunks are set up as MPLS LSPs between selected source-destination pairs. We assume a centralized route server that computes the path for each trunk request using one of the routing algorithms detailed in Section 5.2. The experimental setup used in evaluating these algorithms will be discussed in Section 5.3.
5.2 Routing Algorithms

The IP network is characterized by a sparsely connected topology. Therefore traditional path routing algorithms used in densely connected circuit switched topologies [35] like Direct Path Only (DPO), Success to the Top (STT), State-Dependent Routing (SDR) and Approximate State-Dependent Routing (ASDR) are not appropriate. Most paths between source-destination pairs in IP networks have to traverse multiple links.

Kodialam and Lakshman listed several requirements for a path selection algorithm for MPLS [28]. These include: (1) Necessity to use online algorithms, (2) Use of knowledge of ingress-egress points of LSPs, (3) Good re-routing performance upon link failure, (4) Routing without traffic-splitting, (5) Low computational requirements, and (6) Feasibility of distributed implementation.

Typically shortest path based algorithms are often used in identifying suitable paths for routing. The link-state based routing protocols like the Open Shortest Path First (OSPF) assign a weight to each link and then select the route along the path with the least weight between a source-destination pair. If capacity is not available along this path, then the trunk setup request is blocked.

Several alternate path routing algorithms that use more sophisticated QoS-sensitive routing policies have been proposed. In these algorithms, alternate paths are explored if the shortest path based on OSPF fails. Because of the irregular and sparse topology of the IP network, these algorithms need to intelligently consider alternate paths. These algorithms use the residual bandwidth along each link to decide on feasible paths between a source-destination pair. In our study we concentrate on different
routing algorithms derived from [28] and [35]. The algorithms to be examined include Min-Hop [22], WAPF [35] and MIRA [28].

We propose another interesting algorithm which is an extension of the Min-Hop algorithm, called the Extended Min-Hop (Ext-Min-Hop). This algorithm selects the path with the least weight where the weight of each link is based on statistics of all the LSPs already set up. Each time a path is set up, the weights of all the links along that path are increased. It aims at reducing the possibility of using highly weighted links since these links are “critical” in the sense that more LSPs tend to use them. The Ext-Min-Hop algorithm works as follows:

1. Find the maximal link propagation delay $Max_{propagation\_delay}$ among all the links in the network topology;

2. For each link $l(i, j)$, set its initial weight $weight(i, j) = propagation\_delay(i, j)/Max_{propagation\_delay}$;

3. For each link $l(i, j)$, set its weight incremental value $add_{weight}(i, j) = propagation\_delay(i, j)/Max_{propagation\_delay}$;

When an LSP set-up request $(s, t, B)$ comes,

4. Remove the links with residual bandwidth less than $B$ from the network;

5. Select the path with the least weight between $s$ and $t$ in the reduced graph;

6. For each link $l(i, j)$ along the path, set $weight(i, j) = weight(i, j) + add_{weight}(i, j)$.

Here $add_{weight}(i, j)$ is the increment that is added to the link whenever it is used in one of the paths between a source and a destination. As more LSPs choose the link $l(i, j)$ to be included in their paths, the weight of the link increases correspondingly. This increase will make the link less attractive to be included in future LSPs.
5.3 Network Configuration

Two different network topologies were used to compare the different routing algorithms. The first topology, adopted from [28], consists of a 15-node topology shown in Figure 5-1(a). In this topology all the links are bi-directional. There are two different kinds of links in the network. The darker links have four times the capacity of the light links. A subset of the nodes in the network act as the ingress-egress pairs $S_i-D_i$. In this topology, four such pairs are considered.

The second topology, adopted from [35], is presented in Figure 5-1(b). It consists of 12 core routers and 42 links. Each link has the same capacity and any pair of nodes can act as the ingress-egress pair. This configuration was selected to especially study the performance of MIRA in a situation where any pair of routers can act as ingress-egress pairs.

VoIP MPLS trunk setup requests are generated randomly between any source-destination pair. The source-destination pair is randomly picked from all possible ingress-egress pairs, with each pair having the same probability of being
chosen. Each trunk request has a bandwidth demand that is uniformly distributed
between 1 and 4 units. The same parameters were used in [28]. When the request is
generated, the network attempts to set up the corresponding LSP.

We consider two performance metrics to measure the performance of different
algorithms. The first metric is the probability of blocked request. This is computed as
the ratio of the number of blocked requests to the total number of LSP setup requests
that are generated. The second metric that we consider is the distribution of the
propagation delays for the paths generated by the algorithms. If all the link
propagation delays are constant, then this metric is an alternative measure for the
number of hops in each path.

We conduct several simulation experiments for each configuration, each time with a
different random seed. The results presented in the next section are the average of
multiple trials.

5.4 Performance Results

Our first set of results are presented in Figure 5-2 for Topology 1. In Figure 5-2(a), the
percentage of blocked calls is plotted as a function of the link capacity. We notice that
with increasing link capacity the percentage of blocked calls reduces consistently for
all the algorithms, as expected. We further notice that OSPF, which considers only the
shortest paths, exhibits the worst performance among all the algorithms. All the other
algorithms that consider alternate path show vast improvement compared to OSPF.
Figure 5-2(b) presents the percentage of blocked calls as a function of the total number
of calls. We notice that the percentage of blocked calls increases with increase in the
number of calls, as expected. Once again, OSPF performs the worst while the rankings
of other algorithms are similar to those in Figure 5-2(a).

Corresponding results for Topology 2 are presented in Figure 5-3. Same as in Topology 1, the percentage of blocked calls increases with increase in link capacity or the number of calls. However we notice that the rankings of other algorithms change compared to Topology 1 except that OSPF still performs the worst. This shows that the routing performance of the algorithms is topology dependent. Locations of potential source-destination pairs interfere with the performance of the algorithms.
(a) Percentage of Blocked Calls as Function of Link Capacity – 4000 Calls

(b) Percentage of Blocked Calls as Function of Number of Calls – 1080 Kbps

Figure 5-2: Performance Results for Topology 1
(a) Percentage of Blocked Calls as Function of Link Capacity – 4000 Calls

(b) Percentage of Blocked Calls as Function of Number of Calls – 480 Kbps

Figure 5-3: Performance Results for Topology 2
We summarize our observations as follows:

- When the link capacity is small or the number of calls is large, the performance of all the algorithms (except OSPF) are close since the network is saturated and a high percentage of the calls received after the saturation point are blocked.

- As link capacity increases or the number of calls reduces, the differences in performance between the different algorithms become more apparent.
  - We see that MIRA performs the best for Topology 1 and the second best for Topology 2. This is because MIRA exploits the knowledge of ingress-egress pairs to reduce the impact of current routing on future demands. It takes into account “critical links” for ingress-egress pairs and attempts to first consume non-critical links as much as possible so that the interference to satisfy future demands is reduced. However since it uses much more knowledge of network state, the cost of computation is high.
  - WAPF performs very close to MIRA for Topology 1 but exhibits poor performance for Topology 2. This is due to the difference between the distributions of potential source-destination pairs in the two topologies. In Topology 1, paths for each \( s-d \) pair share very few common links. WAPF chooses path alternately among all the possible paths for each \( s-d \) pair so that it avoids using common links of \( s-d \) pairs to some extent. For Topology 2, each node has the same probability of being chosen as the source or the destination node. The paths between an \( s-d \) pair tend to have small number of hop counts, and many shortest paths are just direct links between \( s \) and \( d \). For a given \( s-d \) pair, WAPF selects paths alternately but the longer the path, the more \( s-d \) pairs it will interfere with, i.e., it reduces the residual capacity of more potential paths so that more future calls will be blocked.
Min-Hop performs the second worst for Topology 1 since it keeps using the shortest path of each s-d pair until it is used up and then tries to find an alternate path. If the current shortest paths of several pairs share common links then setting up a path for one pair in fact decreases the max-flow of other pairs. Since Min-Hop keeps doing this, it causes more future calls to be blocked. But for Topology 2, many shortest paths are only direct links which interfere much less with other pairs. So the performance of Min-Hop is better than WAPF in Topology 2.

For Topology 1, Ext-Min-Hop acts like a combination of Min-Hop and WAPF, so its performance is in between the two. For Topology 2, Ext-Min-Hop performs the best among all the algorithms.

When the link capacity is large or the number of calls is small, a path can be found for most of the calls. Hence, the percentage of blocked calls for all the algorithms is close to zero.

Figure 5-4 summarizes the propagation delay distribution for both topologies. Since we assume that all the links have the same propagation delay (we assume a propagation delay of 10ms for each link), the results presented in the figure are also an indirect reflection on the hop-count of the paths setup by the different algorithms. From the figure, we notice that:

- OSPF performs best since it only uses the shortest paths and doesn’t find alternate paths.
- WAPF performs the worst because it selects from all the possible paths alternatively. It tends to choose a larger number of long paths than other algorithms and so the delay of a path selected tends to be large.
• The behaviors (presented by the curves in the figure) of the other three algorithms, Min-Hop, MIRA and Ext-Min-Hop, are affected by the topology, the locations of ingress-egress pairs and the number of requests accepted. Min-Hop exhibits very good performance for Topology 1 while Ext-Min-Hop performs well for Topology 2.

5.5 Summary

In this chapter we have considered the performance of several routing algorithms in setting up MPLS LSPs for voice traffic in an IP network. The performance of these different algorithms was examined using two different network topologies. Performance measures considered include the probability of blocked calls and the propagation delay distribution for the paths. It was found that OSPF performs worst, because it considers only shortest paths in satisfying a path setup request. Other algorithms which examine alternate paths show good improvement over OSPF. MIRA in particular exhibits very good performance in both topologies. We have also introduced a variation of Min-Hop algorithm based on increasing the weight of used links. The new variation improves performance to some extent over Min-Hop depending on the topology. Further investigations are needed to determine the effects of the topology and the location of source-destination pairs on performance. Also suitable algorithms for combined off-line and on-line traffic engineering need to be investigated.
Figure 5-4: Percentage of Calls as Function of Propagation Delay for 4000 Calls
Chapter 6

New Bandwidth-Delay-Constrained QoS
Routing Algorithms

In this chapter, we present new bandwidth-delay constrained routing algorithms that make use of the knowledge of the ingress-egress node pairs in the network to achieve good performance in terms of reducing the rejection rates for setting up new paths. These algorithms are of interest in the context of setting up paths for VoIP flows over the Internet characterized by stringent bandwidth and delay constraints. Simulation is used to evaluate our algorithms and compare their performance against some existing algorithms for bandwidth constraints that have been modified to handle delay constraints. The results show that our proposed algorithms outperform the others under a wide range of workload, topology and system parameters.

6.1 Problem Definition

We study the bandwidth-delay-constrained routing problem in the MPLS domain. The network consists of $n$ routers. A subset of these routers are assumed to be ingress-egress routers between which paths can be set up. Each link in the network has two properties: residual bandwidth and delay. The residual bandwidth is defined as the difference between the link bandwidth and the sum of the bandwidths of all the paths already assigned to that link. The delay of a link consists of the link propagation delay
and the queuing delay at the starting node. We use source or server-based routing strategy. Both of these strategies are simple and can guarantee loop-free routes. We assume that information concerning ingress-egress nodes is known and changes occur infrequently. For computing the explicit route, whether for source or server-based routing, the ingress router or the route server needs to know the current network topology, link residual bandwidth and delay. We assume that the information is either known or that a link state routing protocol is used to acquire the information.

We define below some of the notations to be used in this chapter. The given network is modeled as a directed graph $G(V, E, P)$. $V$ is the set of nodes (routers) and $E$ is the set of edges (links) as defined in Chapter 2. Each link $l(i, j) \in E$ is associated with a vector $(b(i, j), d(i, j))$, where $b(i, j)$ is the residual bandwidth and $d(i, j)$ is the delay of link $l(i, j)$. $P = P_i \times P_e$ is considered as the set of potential ingress-egress pairs, where $P_i \subset V$ is the set of ingress routers and $P_e \subset V$ is the set of egress routers. We denote a generic element of this set $P$ by $(s, t)$ with $s \in P_i$ and $t \in P_e$. All path setup requests are assumed to occur between these pairs. Let $p$ denote the total number of pairs. Let $n$ represent the number of routers and $m$ the number of links in the network. The setup request for path $i$ is defined by a quadruple $(s_i, t_i, B_i, D_i)$, where $s_i$ specifies the ingress router, $t_i$ specifies the egress router, $B_i \in R_0$ specifies the amount of bandwidth required and $D_i \in R_0^+$ specifies the delay requirement.

We assume that path setup requests arrive one at a time and there is no prior knowledge of future requests. The objective is to determine one feasible path for each request. We use the call blocking ratio as the performance metric, to compare the different algorithms:

$$call\ blocking\ ratio = \frac{\text{number of requests rejected}}{\text{total number of requests}}.$$
The optimization goal is to minimize the *call blocking ratio*, i.e., to maximize the number of requests accepted into the network.

### 6.2 Key Principles

In this section, we present the key ideas used in our routing algorithms. Our objective of path selection is to accept as many requests as possible into the network. Whether a request can be accepted depends on whether sufficient resource can be found to satisfy both the bandwidth and delay requirements of the request. Therefore we should conserve as much resource as possible that would be “critical” to meet future demands. Although we do not assume any knowledge of future requests, the routers where paths can potentially originate and terminate are known since these are the network’s ingress and egress routers. This information is taken into consideration to determine the criticality of resource on the links. Details are given in the following subsections.

#### 6.2.1 Delay-Weighted Capacity

Consider an ingress-egress pair \((s, t)\). Let us imagine a path between \(s\) and \(t\) as a kind of “machine” to accommodate requests with bandwidth and delay constraints. We can measure the “power” of the machine by the end-to-end delay of that path. The smaller the end-to-end delay, the more powerful the machine since the machine can satisfy requests with tight delay requirements. The number of requests the machine can accommodate, called the “capacity” of the machine, is measured by the bandwidth of the path. It is easy to see that the most “powerful machine” for the pair \((s, t)\) is the least delay path between \(s\) and \(t\). We can then remove from the network all links used by the most powerful machine, which means that we eliminate the links belonging to the least delay path. The second most powerful machine for the pair \((s, t)\) is the least delay
path computed in the remaining graph. If this process is repeated until no path exists
between $s$ and $t$, we get a set of least delay paths represented by

$$LP_{st} = \{LP^1_{st}, ..., LP^i_{st}, ..., LP^{k-1}_{st}\}.$$ $LP^i_{st}$ is the least delay path computed in the graph where

the links belonging to $LP^1_{st}$, ..., $LP^{i-1}_{st}$ are eliminated. Let $B^i_{st}$ denote the residual

bandwidth of $LP^i_{st}$ and $D^i_{st}$ the end-to-end delay. The total number of paths in $LP_{st}$ is

represented by $k_{st}$.

An illustrative example is shown in Figure 6-1. In Figure 6-1(a), the first least delay
path between $s$ ($s=0$) and $t$ ($t=5$) is computed, which is $P_1(0, 2, 5)$ with bandwidth 5
and delay 3. Path $P_1$ is abstracted as a dotted link from $s$ to $t$ shown in Figure 6-1(b).
The second least delay path $P_2(0, 3, 5)$ is then calculated in Figure 6-1(b) and
abstracted as a link in Figure 6-1(c). The process repeats until no path can be found as
shown in Figure 6-1(d), where the network is abstracted as a set of dotted links
between $s$ and $t$ representing the set of least delay paths computed.

By the definition of $LP_{st}$, we see that priority is put on the delay constraint. The reason
is as follows. We know that delay is an additive metric and the end-to-end delay of a
path is the sum of the link delays on that path. So the "power" of a path with respect to
delay depends on the whole set of links along that path. Thus the granularity to
measure the capability of an ingress-egress pair to satisfy delay constraint is the path.
On the other hand, bandwidth is a concave metric and the bandwidth of a path is
determined by the minimum link bandwidth along that path. The minimum bandwidth
links are called bottleneck links. The "power" of a path with respect to bandwidth only
relies on the bottleneck links. So the granularity to measure the capability of an
ingress-egress pair to satisfy bandwidth constraint is the link. This finer granularity
gives higher flexibility for path selection between an ingress-egress pair.

We consider maximizing the sum of machine capacity for each ingress-egress pair \((s, t)\). Each machine capacity is weighed by the power of that machine which indicates the importance of its capacity. We define the delay-weighted capacity (DWC) for each pair \((s, t)\) as in Definition 7.

![Diagram](image)

Figure 6-1: Illustrative Example.

Vector associated with each link is (bandwidth, delay).

**Definition 7:** The delay-weighted capacity (DWC) of an ingress-egress pair \((s, t)\) is defined as a weighted sum of the bandwidth of the paths in the set \(LP_{st} = \{LP_{st}^1, LP_{st}^2, ..., LP_{st}^{k_s} \}\). The weights are inversely proportional to the end-to-end delay values of the associated paths.

\[
DWC_{st} = \sum_{LP_{st} \in LP_{st}} \frac{B_{st}^i}{D_{st}^i}
\]

With the definition of DWC given above, an *optimal path* for a request can be computed as the route that maximizes the weighted sum of the DWC value between
every ingress-egress pair after the route has been set up.

6.2.2 Critical Link

If we route a request on a bottleneck link of any least delay path in the set $LP_{st}$, the DWC value of the pair $(s, t)$ decreases. Such a link is defined as a “critical link” for pair $(s, t)$. It has the property that whenever a path is routed over it, the DWC value of one or more ingress-egress pairs decreases. We represent the set of critical links for the pair $(s, t)$ by $C^i_{st} = \{C^1_{st}, \ldots, C^i_{st}, \ldots, C^n_{st}\}$, where $C^i_{st}$ consists of all the bottleneck links for the least delay path $LP^i_{st}$.

It is clear we should avoid routing paths on critical links as much as possible. We do this by assigning weights to critical links that are an increasing function of their “criticality”. An extended Dijkstra or Bellman-Ford algorithm can then be used to compute the least weight path as the explicit route for the request.

6.3 Routing Algorithms

6.3.1 Maximum Delay-Weighted Capacity Routing Algorithm

We wish to route a request along a path which preserves the maximal leftover weighted sum of DWC of each ingress-egress pair, i.e., a path that causes the least decrease of the weighted sum after it has been set up. We achieve this by determining appropriate weights for the links in the network and route the request along the least weight path. The weights reflect the “criticality” of the links. Therefore the problem of computing the weights of the links is reduced to one of determining the set of critical
links for all ingress-egress pairs. This can be solved by the iterative process of calculating $LP_{it}$ for each pair $(s, t)$. We can use Dijkstra algorithm to compute the least delay path $LP'_{it}$ at each round in the iteration, which takes $O(n \log n)$. Critical links for $LP'_{it}$ can be found in $O(n)$ since $LP'_{it}$ contains at most $(n-1)$ links. For each ingress-egress pair, the iteration takes at most $O(m)$ rounds since at least one link is eliminated from the graph at each round. The topology of a wire Internet network can be expressed using a planar graph. For this type of networks, we can further reduce the average number of rounds to $O(1)$ as follows. We know from computational geometry [8] that the average degree $d$ of a node in a planar graph is 6. We know that any path between two nodes includes a link originating from the source and an incoming link to the destination. Each time we eliminate the links along the least delay path between an ingress-egress pair, we decrease the outgoing degree of the ingress node and the incoming degree of the egress node by 1. On average, after 6 rounds, no path can be found between that pair of ingress-egress nodes. Therefore for each ingress-egress pair, it takes $O(n \log n + n)$ to compute $LP_{it}$ and find all critical links. Since there are a total of $p$ ingress-egress pairs, the complexity is $O(p(n \log n + n)) = O(np \log n)$.

Once all the critical links are determined, we assign weights to the critical links and route the request along the least weight path. Before doing this, we first eliminate all links with residual bandwidth less than the bandwidth requirement so as to ensure that any path computed in the remaining graph will satisfy the bandwidth constraint. To guarantee that the delay constraint is also satisfied, we use an extended Dijkstra (EDSP) or Bellman-Ford (EBF) algorithm [15] to compute the least weight path. EDSP and EBF can both solve the two-additive-constrained routing problem. Therefore we can ensure the least weight path computed by EDSP or EBF is feasible in terms of delay. We call such a path the delay-constrained least-weight path. The
complexity of ESDP is $O(x^2n^2)$ and the complexity of EBF is $O(xmn)$ where $x$ is a positive integer.

We assume the current request is between routers $a$ and $b$ with demands of $B$ units of bandwidth and $D$ units of end-to-end delay. At this point other requests may already have been routed and the residual capacities of the links have been updated to reflect these allocations. The routing algorithm is detailed below, where $\alpha^i_{st}$ is a property associated with the least delay path $LP^i_{st}$. The selection of the value of $\alpha^i_{st}$ will be discussed following the algorithm.

**Maximum Delay-Weighted Capacity Routing Algorithm (MDWCRA)**

1. Compute the delay-weighted capacity (DWC) values for all $(s,t) \in P$.
2. Compute the set of critical links $C_{st}$ for all $(s,t) \in P$.
3. Compute the link weights
   \[
   w_l = \sum_{(s,t) \in C_{st}} \alpha^i_{st} \quad \forall l \in E
   \]
4. Eliminate all links that have residual bandwidth less than $B$ and form a reduced network.
5. Using the EDSP or EBF algorithm, compute the delay-constrained least-weight path in the reduced network using $w_l$ as the weight on link $l$.
6. Route the request from $a$ to $b$ along this delay-constrained least-weight path and update the residual capacities of the network.

With $\alpha^i_{st}$ selected to be different values, the definition of link weight can assume different meanings:
1. \( w_i = \sum_{(s,t) \in C_u} 1 \), with \( \alpha_{st} = 1 \). The weight of link \( l \) represents the number of ingress-egress pairs for which link \( l \) is critical.

2. \( w_i = \sum_{(s,t) \in C_u} \frac{1}{D_{st}^i} \), with \( \alpha_{st}^i = \frac{1}{D_{st}^i} \). The link weight is inversely proportional to the end-to-end delay value of the least delay path for which link \( l \) is considered critical.

3. \( w_i = \sum_{(s,t) \in C_u} \frac{1}{B_{st}^i \times D_{st}^i} \), with \( \alpha_{st}^i = \frac{1}{B_{st}^i \times D_{st}^i} \). The link weight is inversely proportional to the product of the bandwidth and the end-to-end delay of the least delay path for which link \( l \) is considered critical.

Now we analyze the time complexity of MDWCRA. As discussed above, Step 1 and 2 take \( O(np \log n) \). Step 3 can be piggybacked on Step 2. So no additional complexity is introduced. Step 4 costs \( O(m) \). Step 5 takes the same amount of time as the EDSP or EBF algorithm. The total time complexity is therefore \( O(np \log n + m + x^2 n^2) = O(np \log n + x^2 n^2) \) by EDSP or \( O(np \log n + m + xmn) = O(np \log n + xmn) \) by EBF.

6.3.2 A Simpler Version of MDWCRM

A simpler version of MDWCRM is described as follows with \( \alpha_{st} \) being a property associated with the least delay path \( lp_{st} \) of pair \((s,t)\).

MDWCRM SIMP

1. Compute the least delay path \( lp_{st} \) for all \((s,t) \in P\).

2. Compute the link weights
\[ w_i = \sum_{(s,t) \in \Phi_i} \alpha_{st} \quad \forall l \in E. \]

3. Eliminate all links that have residual bandwidth less than \( B \) to form a reduced network.

4. Using the EDSP or EBF algorithm, compute the delay-constrained least-weight path in the reduced network using \( w_i \) as the weight on link \( l \).

5. Route the request from \( a \) to \( b \) along this delay-constrained least-weight path and update the network's residual capacities.

The difference between this simpler version of the algorithm and the original one is that we compute only one least delay path for each ingress-egress pair instead of a set. We assign weights to the links with the objective of avoiding using those links that belong to the least delay path of any ingress-egress pair. Once the weights are determined, we follow the same steps as in the original MDWCRA. The order of complexity of MDWCRA SIMP is the same as MDWCRA.

6.4 Network Configuration

Two different network topologies were used to compare the different routing algorithms. The first topology consists of 15 nodes as shown in Figure 6-2(a). We call it the MIRA topology to indicate that this topology is adopted from [28] where the MIRA algorithm was introduced. In this topology all the links are considered bi-directional. There are two different kinds of links in the network. The light links have capacity of 12 units and the dark links have capacity of 48 units (taken to model the capacity ratio of OC-12 and OC-48 links). We scale down all capacities by 20 to permit us to experiment with a larger number (thousands) of path set-ups. The delay values of the links are uniformly distributed in the range [0...50ms]. A subset of the
nodes in the network act as the ingress-egress pairs $S_i - D_i$ that are potential source-destination pairs of path setup requests. In the MIRA topology, four such pairs are considered.

The second topology, adopted from [15], expands the major circuits in ANSNET [18] by inserting additional links to increase the connectivity. This topology, called expanded ANSNET topology, consists of 32 nodes and is presented in Figure 6-2(b). Each link has the same capacity of 12 units. The delay values of the links are uniformly distributed in the range [0...50ms]. Five $S_i - D_i$ pairs in this topology are considered as ingress-egress pairs.

![MIRA Topology](image-a) ![Expanded ANSNET Topology](image-b)

(a) MIRA Topology  
(b) Expanded ANSNET Topology

Figure 6-2: Network Topologies

We also examine the performance of the different algorithms in both the MIRA topology and the expanded ANSNET topology with no specified ingress-egress pairs. This means that any pair of the nodes can act as the source-destination pair. These two configurations were selected to study the performance of MDWCRA in a situation where any pair of routers can act as source-destination pairs.
We consider the cases of both evenly and unevenly distributed traffic load in the above four network configurations. For an evenly distributed traffic load, path setup requests are generated randomly between any source-destination pair. The source-destination pair is randomly picked from all possible ingress-egress pairs, with each pair having the same probability of being selected. For the uneven load, a high percentage of traffic is concentrated among a selected subset of ingress-egress pairs. These heavy-loaded pairs are explicitly specified in the MIRA topology and the expanded ANSNET topology with specified ingress-egress nodes. In the case where no ingress-egress nodes are specified, the heavy-loaded pairs are randomly selected from all possible ingress-egress pairs.

Each request has a bandwidth and a delay requirement. The bandwidth requirement is uniformly distributed between 1 and 5 units. The delay requirement is generated from different ranges, viz., [50...65ms], [75...90ms], [100...115ms], [125...140ms] and [150...165ms]. The smaller the values of the delay range, the tighter the delay constraint of a request. Within each range, the value of a delay requirement is uniformly distributed.

We use the call blocking ratio, defined earlier, as the performance metric to compare the performance of the different algorithms. A number of simulation experiments were conducted for each configuration, each time with a different random seed. The results presented in the next section are the average of multiple runs.

6.5 Performance Results

Before we present the results, the set of algorithms examined are described. We modified some of the existing algorithms that consider only bandwidth constraints, to
also consider delay constraints, so that the comparison is more meaningful. The algorithms considered are as follows:

1. the Wang et. al. algorithm with complexity of $O(n \log n + m)$ [45];

2. the delay-constrained Min-Hop (DC Min-Hop) algorithm with complexity of $O(x^2n^2)$ or $O(xmn)$;

3. the delay-constrained WAPF (DC WAPF) algorithm with complexity of $O(x^2n^2)$ or $O(xmn)$;

4. the delay-constrained MIRA (DC MIRA) algorithm with complexity of $O(pn^2\sqrt{m} + pm^2 + x^2n^2)$ or $O(pn^2\sqrt{m} + pm^2 + xmn)$;

5. MDWCRA with complexity of $O(np \log n + x^2n^2)$ or $O(np \log n + xmn)$.

The original Min-Hop [22], WAPF [35] and MIRA [28] algorithms cater to the case of setting up bandwidth-guaranteed connections between a given source and destination. All of these three algorithms utilize the well-known Dijkstra or Bellman-Ford algorithm to calculate the feasible path. We modify them by using the extended Dijkstra (EDSP) or Bellman-Ford (EBF) algorithm. As mentioned in Section 6.3, both EDSP and EBF solve two-additive-constrained routing problem. Therefore we can ensure that the path computed by the extended Min-Hop, WAPF and MIRA algorithms satisfy the delay constraint. We call the three extended algorithms the delay-constrained Min-Hop (DC Min-Hop), DC WAPF and DC MIRA algorithms respectively. The complexity of DC Min-Hop and DC WAPF is determined by the complexity of EDSP ($O(x^2n^2)$) or EBF ($O(xmn)$). Using knowledge of ingress-egress nodes introduces additional complexity in the order of $O(pn^2\sqrt{m} + pm^2)$ in DC MIRA and $O(np \log n)$ in MDWCRA. It is clear
that $O(np \log n)$ is much smaller than $O(pn^2 \sqrt{m + m^2})$. As mentioned in Section 5, $x$ is a positive integer which can take the value of $10d_{st}$, where $d_{st}$ is the distance between s and t, for EDSP and EBF to be practical [15]. The order of $O(p)$ for a specific region is usually small, of the order of $O(x)$. Therefore $O(np \log n)$ is much smaller than $O(x^2 n^2)$ and $O(xmn)$, which makes the total cost of MDWCRA just slightly higher than that of DC Min-Hop and DC WAPF.

In Section 3 of this chapter, we introduced three definitions of link weight for MDWCRA algorithm. After conducting several simulation runs for various configurations, we found that the performance of MDWCRA under the three definitions is very close. For clarity of presentation, we selected MDWCRA and MDWCRA SIMP under link weight definition 3¹ as the representatives of the whole set of algorithms. Hereafter in this thesis when we refer to a path to be setup, we mean a bandwidth-delay constrained path.

Our first set of results is presented in Figure 6-3(a) for evenly distributed traffic load and Figure 6-3(b) for unevenly distributed traffic load. In these figures, the call blocking ratio is plotted as a function of the number of requests. The MIRA topology was used with specified ingress-egress nodes. For the even load, the requests were uniformly generated between the four source-destination pairs. For the uneven load, 80% of the requests were distributed between the two pairs ($S_0, D_0$) and ($S_1, D_1$). We examined the performance of the Wang et. al. algorithm, DC Min-Hop, DC WAPF, DC MIRA, MDWCRA for these configurations with bandwidth constraint $B \in [1...5]$ and delay constraint $D \in [50...65ms], [75...90ms], [100...115ms], [125...140ms], [150...165ms]$ respectively. We found that with increasing number of requests the call blocking ratio increases consistently for all the algorithms, as expected. We also
noticed that the performance ranking of these algorithms by the call blocking ratio remain the same independent of $D$. For clarity we have selected to display cases with $D \in [100\ldots115\text{ms}]$ only in this section.

We notice that MDWCRA exhibits the best performance among all the algorithms. We further observe that when the number of requests is small, the performance of all the algorithms is close. This is because with light load, most of the requests can be accommodated, so the difference in call blocking ratio between the algorithms is small. This is also true when the number of requests is large since the network is saturated and a high percentage of the requests received after the saturation point are blocked. The difference in call blocking ratio is the most apparent under medium load. We magnified this region of Figures 6-3(a) and 6-3(b) and show them in Figures 6-4(a) and 6-4(b) respectively to better illustrate the performance difference among the different algorithms. For network service providers and users, a network with low blocking ratio is of the most interest. Figures 6-5(a) and 6-5(b) magnify the region of Figures 6-3(a) and 6-3(b) with blocking ratio below 30%. In all cases of these figures, the maximum blocking ratio is no more than 5% worse than the average value.

We see that in most cases the Wang et. al. algorithm performs the worst for both even and uneven workload. This algorithm keeps using the least delay path of each $S$-$D$ pair until it is used up and then tries to find an alternate path. The least delay path is obviously the best one to satisfy the delay requirement. But we think of a path as good as the best one as long as it can accommodate the delay constraint. Unnecessarily using the least delay path in fact reduces the chance of accepting future requests with tight delay constraints. Furthermore, the links along the least delay path of one ingress-egress pair tend to have small link delays. Thus these links have high
probability of being on the least delay paths of other ingress-egress pairs. Therefore routing a request along the least delay path of one pair may also reduce the bandwidth of the least delay path of some other pairs. This tends to decrease the number of requests with tight delay constraints to be successfully routed between those pairs. Finally, the least delay path may be long in terms of the hop count and hence consumes more resource in the network.

The DC MIRA algorithm performs very close to the Wang et. al. algorithm for both evenly and unevenly distributed workload. Although DC MIRA utilizes the knowledge of ingress-egress pairs to identify critical links for each pair, the critical links are defined only in terms of bandwidth. Obviously, this consideration does not fit well in the context of setting up bandwidth-delay-guaranteed connections, where the delay criterion plays an important role. The critical links in terms of bandwidth are not necessarily critical for delay. Therefore, protecting these critical links cannot effectively conserve critical resource between ingress-egress pairs to satisfy future requests. On the contrary, it may decrease the chance of future requests being accepted.

The DC Min-Hop algorithm performs the third worst in most cases among the algorithms studied. It finds an alternate path when the shortest path of each $S-D$ pair is used up. The rationale of selecting the shortest path is that it uses less resource in the network. However, unlike the DC MIRA algorithm, DC Min-Hop does not consider the information of ingress-egress nodes. The links along the shortest path, if heavily loaded, may make it impossible to satisfy future requests between certain ingress-egress pairs. Since DC Min-Hop keeps using these links, it causes more future calls to be blocked than the average algorithm.
The DC WAPF algorithm performs in the middle of all the algorithms. For the case of even load, DC WAPF performs much better than the above three algorithms. The reason lies in the way path selection is made. Since DC WAPF always chooses the widest available path, when the variation of link bandwidth in the network is small, the algorithm alternately chooses paths for each ingress-egress pair. With an evenly distributed load, the attempt of load-balancing the network traffic is effective as seen in Figure 6-4(a). When the traffic load is unevenly distributed (Figures 6-4(b) and 6-5(b)), the performance of DC WAPF is not as good compared to the case of even load. This shows that balancing the traffic while the actual workload is not even does not help to improve the performance much.

The algorithm that performs the best for both distributions of workload is the MDWCRA algorithm with the former slightly better than the latter. Both algorithms attempt to first use links that are not crucial to future requests for each ingress-egress pair. Critical links are identified with priority on delay criterion. By deferring loading of the critical links, the potential of each ingress-egress pair to satisfy future delay constraints is effectively preserved. Moreover, the weights of the links in the algorithms are assigned according to their “criticality”. The computation of least weight path maximizes the number of future requests accepted.
Figure 3: Call blocking ratio as function of the number of requests

MIRA topology with specified ingress-egress nodes, $B \in [1...5]$, $D \in [100...115\text{ms}]$
(a) Evenly distributed traffic

(b) Unevenly distributed traffic

Figure 4: Call blocking ratio as function of the number of requests

MIRA topology with specified ingress-egress nodes, $B \in [1...5]$, $D \in [100...115\text{ms}]$
Figure 5: Call blocking ratio as function of the number of requests

MIRA topology with specified ingress-egress nodes, $B \in [1 \ldots 5]$, $D \in [100 \ldots 115ms]$
Figure 6: Call blocking ratio as function of the number of requests
Expanded ANSNET topology with specified ingress-egress nodes, $B \in [1\ldots5]$, $D \in [150\ldots165\text{ms}]$
Figure 7: Call blocking ratio as function of the number of requests
Expanded ANSNET topology with specified ingress-egress nodes, $B \in [1...5], D \in [150...165ms]$
Figure 8: Call blocking ratio as function of the number of requests
MIRA topology with non-specified ingress-egress nodes, $B \in [1...5], D \in [100...115\text{ms}]$
Figure 9: Call blocking ratio as function of the number of requests
Expended ANSNET topology with non-specified ingress-egress nodes, $B \in [1...5], D \in [125...140ms]$
Figures 6-6 and 6-7 present the call blocking ratio distribution of all the algorithms using the expanded ANSNET topology with specified ingress-egress nodes under both even load (Figures 6-6(a) and 6-7(a)) and uneven load (Figures 6-6(b) and 6-7(b)) cases. For the even load case, the requests are uniformly generated between the five source-destination pairs. For the uneven load case, 80% of the requests are distributed between the two pairs \((S_0, D_0)\) and \((S_3, D_3)\). The ranges of bandwidth and delay constraints are \(B \in [1...5]\) and \(D \in [150...165\text{ms}]\) respectively. We notice that the performance behaviors (as shown by the shape of the curves) of all the algorithms are similar with those in the MIRA topology with specified ingress-egress nodes.

Figures 6-8 and 6-9 summarize the call blocking ratio as a function of the number of requests for the MIRA topology and the expanded ANSNET topology without specified ingress-egress nodes. Figures 6-8(a) and 6-9(a) present the performance of the algorithms with an evenly distributed workload. The source-destination pair of a request is randomly selected from all the nodes. Figures 6-8(b) and 6-9(b) show the performance with an uneven load where 64% of the traffic distributed in four source-destination pairs. These four heavily loaded pairs are uniformly selected from all the nodes. The bandwidth and delay constraints for these simulation experiments are shown in the figures.

We notice that the performance of the algorithms in topologies with non-specified ingress-egress nodes is somewhat different from that in topologies with explicit ingress-egress nodes. In a topology where each node acts as a potential source or destination, each link in the network becomes crucial since the end points of that link form a potential source-destination pair. This means that each link has the property that whenever a path is routed over that link, the number of requests that can be
accepted between one or more ingress-egress pairs decreases. Therefore, the performance of an algorithm depends on how it distributes the paths.

For the evenly distributed workload shown in Figures 6-8(a) and 6-9(a), we see that the Wang et. al. algorithm performs the worst and the MDWCRM the best. When the load is light, the performance of all the algorithms (except the Wang et. al.) is similar since most requests can be accommodated except those with very tight delay constraints. Under a medium load the behaviors of the algorithms are similar with those in topologies with explicit ingress-egress. When the load gets heavy, the DC Min-Hop algorithm exhibits better performance than the other algorithms except MDWCRM. This is because DC Min-Hop uses the path with the least number of hops which affects fewer number of potential source-destination pairs. DC Min-Hop attempts to retain as much resource as possible in the network. Therefore, when the workload is heavy, this strategy is effective. The performance of MDWCRM is always the best in all the workload ranges used in our experiments. When possible ingress-egress pairs are not specified, MDWCRM treats equally all the source-destination pairs formed by the nodes in the network. It performs very well since it is able to protect those links that would affect the most number of potential source-destination pairs.

For the uneven load presented in Figures 6-8(b) and 6-9(b), we notice that the behavior of the algorithms are much like those with an even load but the differences between the algorithms become much smaller. Although MDWCRA is still the best performer, its performance is very close to the performance of the others. This shows that the strategy of treating each possible source-destination pair equally while the actual traffic load is unbalanced does not effectively bring out the advantage of
6.6 Summary

In this chapter, we have presented new algorithms for setting up bandwidth-delay constrained paths, for example, MPLS paths (LSPs) for voice traffic in an IP network. The algorithms exploit the knowledge of ingress-egress nodes in making path selection taking into consideration both the bandwidth and delay criteria. These algorithms route paths based on the notion of “delay-weighted capacity” so as to accommodate the maximum number of future requests. Simulation experiments were conducted to examine the performance of the new algorithms using two different network topologies under evenly and unevenly distributed traffic load. The performance measure considered is the call locking ratio. We find that the maximum delay-weighted capacity scheme performs the best in comparison with the Wang et. al. algorithm, DC Min-Hop, DC WAPF, and DC MIRA algorithms, which either do not take the ingress-egress information into account or cater only to the bandwidth requirement. The difference in performance varies with the operating conditions.
Chapter 7

Conclusions and Future Work

7.1 Major Contribution of Thesis

We conclude the major contributions in this thesis as follows.

1. We have studied the performance of several existing routing algorithms for setting up bandwidth-guaranteed MPLS LSPs for voice traffic in an IP network. Simulation experiments were performed with two different network topologies. The performance results show that MIRA exhibits very good performance in both topologies. This indicates that considering the knowledge of ingress-egress pairs in routing algorithms helps to improve performance by optimizing network efficiency.

2. We have proposed new bandwidth-delay constrained routing algorithms taking into account knowledge of the ingress-egress node pairs. Our algorithms are based on computing the “delay-weighted capacity” for each ingress-egress pair. We then identify “critical links” as those links whose inclusion in a path will cause the delay-weighted capacity of several ingress-egress pairs to decrease. The routing algorithms try to avoid using the critical links as much as possible by assigning large weights to the critical links as a function of their criticality. An
extended Bellman-Ford algorithm is then used in selecting the path with the least weight for each request. We compare the performance of our algorithms with other algorithms derived by modifying some bandwidth constrained routing algorithms to support delay constraints. The call-blocking ratio, defined as the ratio of the number of rejected requests to the total number of requests, is used as the performance measure. Simulation experiments were run with two different network topologies to compare the performance of the algorithms. The results show that our algorithms, named maximum delay-weighted capacity routing algorithms, outperform the other algorithms under a wide range of workload and system parameters. The performance gain is achieved without a corresponding increase in overhead.

7.2 Future Work

To develop efficient routing algorithms, further investigations are needed to determine the effects of the topology and the location of ingress-egress pairs on performance. Furthermore, the new algorithms proposed define critical links with priority on the delay criterion. The relation and interaction of critical links in terms of multiple metrics like bandwidth and delay need to be investigated further.

Routing must work with other network components in order to provide guaranteed services. The relevant components include maintenance of network state, resource reservation, admission control, QoS negotiation and packet scheduling policies. Design decisions on these components impact the routing performance and cost trade-off of QoS routing. Complete design and implementation of a routing protocol, must also take these factors into account.
References


[8] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf, Computational


