QoS-based multipath routing for the Internet

Bing Chen

University of Nevada, Las Vegas

Follow this and additional works at: https://digitalscholarship.unlv.edu/rtds

Repository Citation
https://digitalscholarship.unlv.edu/rtds/3075

This Dissertation is brought to you for free and open access by Digital Scholarship@UNLV. It has been accepted for inclusion in UNLV Retrospective Theses & Dissertations by an authorized administrator of Digital Scholarship@UNLV. For more information, please contact digitalscholarship@unlv.edu.
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
QOS-BASED MULTI-PATH ROUTING FOR THE INTERNET

by

Bing Chen

Bachelor of Engineering
Liaoning University of Industry, P.R. China
1987

Master of Science
Shenyang Institute of Automation
Chinese Academy of Science, P.R. China
1990

A dissertation submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy Degree
Department of Electrical and Computer Engineering
Howard R. Hughes College of Engineering

Graduate College
University of Nevada, Las Vegas
August 1999
Dissertation Approval
The Graduate College
University of Nevada, Las Vegas

July 16, 1999

The Dissertation prepared by

Bing Chen

Entitled

QoS-based Multi-path Routing For The Internet

is approved in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Engineering

Examination Committee Chair

Dean of the Graduate College

Examination Committee Member

Examination Committee Member

Graduate College Faculty Representative
ABSTRACT

QoS-based Multi-path Routing For The Internet

by

Bing Chen

Dr. Shahram Latifi, Examination Committee Chair
Professor of Electrical Engineering
University of Nevada, Las Vegas

The new generation of network services is being developed for incorporation in communication infrastructure. These services, generally called Quality of Services (QoS), should accommodate data file, video, and audio applications. The different performance requirements of these applications necessitate a re-examination of the main architectural components of today’s networks, which were designed to support traditional data applications. Routing, which determines the sequence of network nodes a packet traverses between source and destination, is one such component. Here, we examine the potential routing problems in future Internet and discuss the advantages of class-based multi-path routing methods. The result is a new approach to routing in packet-switched networks, which is called Two-level Class-based Multi-path routing with Prediction (TCMP). In TCMP, we compute multiple paths between each source and destination based on link propagation delay and bottleneck bandwidth. A leaky bucket is adopted in each router to monitor the bottleneck bandwidth on equal paths during the network’s stable period, and to guide its traffic forwarding. The TCMP can avoid frequent flooding of routing information in a dynamic routing method; therefore, it can be applied to large network topologies.
TABLE OF CONTENTS

ABSTRACT ................................................................. iii
TABLE OF CONTENTS ................................................ iv
LIST OF TABLES ........................................................ vi
LIST OF FIGURES ..................................................... vii
ACKNOWLEDGMENTS ................................................. ix

CHAPTER 1 INTRODUCTION ........................................ 1

CHAPTER 2 INTERNET ROUTING ARCHITECTURE
AND LIMITATIONS ..................................................... 4
  2.1 Internet Routing Architecture Overview ......................... 4
  2.2 Routing Characteristics .............................................. 6
    2.2.1 Distance Estimation .............................................. 7
    2.2.2 Route Computation .............................................. 7
    2.2.3 Information Propagation ...................................... 11
    2.2.4 Packet Forwarding ............................................. 11
  2.3 Requirements for New Services .................................... 11
    2.3.1 Premium Service ................................................ 12
    2.3.2 Assured Service ................................................. 14
  2.4 The Limitations of Current Internet Routing Architecture .... 15
    2.4.1 Single Metric .................................................... 16
    2.4.2 Single Path Routing .......................................... 17
    2.4.3 Single Service .................................................. 17
  2.5 Design Goals of QoS-based Routing ............................... 21
    2.5.1 General Design Goals for Routing Algorithms ........... 21
    2.5.2 Design Goals for Routing to Support New Services .... 22

CHAPTER 3 RELATED WORK AND MOTIVATION ................ 24
  3.1 Constraint-based Routing .......................................... 25
  3.2 Multipath Routing in Packet-Switched Networks .............. 28
  3.3 Routing in Circuit-Switched Networks ............................ 31
  3.4 Routing in High Speed Networks .................................. 34
  3.5 Real-Time Services ................................................. 35
  3.6 Traffic Engineering ................................................ 38
  3.7 Conclusions ......................................................... 40

CHAPTER 4 THE CONSIDERATIONS OF MULTIPLE METRICS
AND MULTIPLE PATHS ............................................... 43
  4.1 Multiple Metrics .................................................... 43
  4.2 Multiple Paths ....................................................... 46

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
<table>
<thead>
<tr>
<th>CHAPTER 5 A TWO-LEVEL CLASS-BASED MULTIPATH ROUTING WITH PREDICTION</th>
<th>53</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Routing Scheme Description</td>
<td>55</td>
</tr>
<tr>
<td>5.2 Neighbor Monitoring Table</td>
<td>57</td>
</tr>
<tr>
<td>5.3 Routing Table Construction</td>
<td>60</td>
</tr>
<tr>
<td>5.4 Packet Forwarding</td>
<td>61</td>
</tr>
<tr>
<td>5.5 Resource Reservation</td>
<td>66</td>
</tr>
<tr>
<td>5.6 Information Distribution</td>
<td>68</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CHAPTER 6 ANALYSIS OF THE TCMP ALGORITHM</th>
<th>69</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1 Loop Freedom in TCMP</td>
<td>69</td>
</tr>
<tr>
<td>6.2 Algorithm Complexity</td>
<td>70</td>
</tr>
<tr>
<td>6.3 Worst-Case Steady-State Delays</td>
<td>72</td>
</tr>
<tr>
<td>6.3.1 Negligible Packet Size</td>
<td>75</td>
</tr>
<tr>
<td>6.3.2 Non-Negligible Packet Size</td>
<td>77</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CHAPTER 7 A SIMULATION STUDY</th>
<th>79</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1 Simulator Design</td>
<td>79</td>
</tr>
<tr>
<td>7.1.1 Traffic Load</td>
<td>80</td>
</tr>
<tr>
<td>7.1.2 Performance Metrics</td>
<td>81</td>
</tr>
<tr>
<td>7.1.3 Updating Mechanism</td>
<td>82</td>
</tr>
<tr>
<td>7.2 The Comparison of Class-based Routing with Other Routing Schemes</td>
<td>83</td>
</tr>
<tr>
<td>7.2.1 Static Routing</td>
<td>84</td>
</tr>
<tr>
<td>7.2.2 Dynamic Routing</td>
<td>86</td>
</tr>
<tr>
<td>7.3 Traffic Performance Using TCMP Scheme in a Small Network</td>
<td>94</td>
</tr>
<tr>
<td>7.3.1 Evenly Distributed Load in Quasi-Static Routing</td>
<td>95</td>
</tr>
<tr>
<td>7.3.2 Unevenly Distributed Load in Quasi-Static Routing</td>
<td>96</td>
</tr>
<tr>
<td>7.3.3 Unevenly Distributed Load in Dynamic Network</td>
<td>98</td>
</tr>
<tr>
<td>7.4 Traffic Performance Using TCMP Scheme in a Large Network</td>
<td>100</td>
</tr>
<tr>
<td>7.5 Supporting Guaranteed Services</td>
<td>103</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CHAPTER 8 CONCLUSIONS AND FUTURE WORK</th>
<th>107</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1 Summary of the Contributions</td>
<td>108</td>
</tr>
<tr>
<td>8.2 Future Work</td>
<td>109</td>
</tr>
</tbody>
</table>

BIBLIOGRAPHY ........................................................................................................ 112

VITA ................................................................................................................................ 122
LIST OF TABLES

Table 4.1 Simulation Results ................................................................. 50
Table 5.1 Neighbor's Credit Calculation Table .................................. 60
Table 5.2 Routing Table ............................................................................. 61
Table 7.1 Source-Destination Pairs of Traffic Flows ......................... 87
Table 7.2 Equal Paths from Source-Destination Pairs ..................... 87
Table 7.3 Unevenly Loaded Links ........................................................... 97
Table 7.4 More Unevenly Loaded Links .................................................. 99
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 2.1</td>
<td>The Internet Routing Architecture</td>
<td>5</td>
</tr>
<tr>
<td>Figure 2.2</td>
<td>The TCP/UDP Function</td>
<td>18</td>
</tr>
<tr>
<td>Figure 3.1</td>
<td>Example of Routing Loops</td>
<td>29</td>
</tr>
<tr>
<td>Figure 4.1</td>
<td>Practical Router Configuration</td>
<td>47</td>
</tr>
<tr>
<td>Figure 4.2</td>
<td>An Example of Relaxing the Best Paths Criteria</td>
<td>48</td>
</tr>
<tr>
<td>Figure 4.3</td>
<td>A Network Topology with Ten Nodes</td>
<td>49</td>
</tr>
<tr>
<td>Figure 5.1</td>
<td>The Router Structure</td>
<td>56</td>
</tr>
<tr>
<td>Figure 5.2</td>
<td>Neighboring Structure of Node $n_k$</td>
<td>58</td>
</tr>
<tr>
<td>Figure 5.3</td>
<td>The Algorithm for Computing Neighbor's Credit</td>
<td>59</td>
</tr>
<tr>
<td>Figure 5.4</td>
<td>Path Calculation Algorithm</td>
<td>62</td>
</tr>
<tr>
<td>Figure 5.5</td>
<td>The Leaky Bucket</td>
<td>63</td>
</tr>
<tr>
<td>Figure 5.6</td>
<td>Using Leaky Bucket for Monitoring the Link Load</td>
<td>64</td>
</tr>
<tr>
<td>Figure 5.7</td>
<td>Packet Forwarding Algorithm</td>
<td>65</td>
</tr>
<tr>
<td>Figure 5.8</td>
<td>Bottleneck Link for Several Destinations</td>
<td>66</td>
</tr>
<tr>
<td>Figure 5.9</td>
<td>Resource Reservation Algorithm</td>
<td>67</td>
</tr>
<tr>
<td>Figure 5.10</td>
<td>Two-link in a Network System</td>
<td>70</td>
</tr>
<tr>
<td>Figure 7.1</td>
<td>Traffic Model</td>
<td>81</td>
</tr>
<tr>
<td>Figure 7.2</td>
<td>The Simulated Six-Node Network</td>
<td>83</td>
</tr>
<tr>
<td>Figure 7.3</td>
<td>The Simulated Six-Node Network with More Degree</td>
<td>84</td>
</tr>
<tr>
<td>Figure 7.4</td>
<td>Packet Drops with Running Time</td>
<td>85</td>
</tr>
<tr>
<td>Figure 7.5</td>
<td>Queueing Delay with Running Time</td>
<td>85</td>
</tr>
<tr>
<td>Figure 7.6</td>
<td>Packet Drops with the Increasing of Updating Interval</td>
<td>88</td>
</tr>
<tr>
<td>Figure 7.7</td>
<td>The Times of Link Utility Over 90 Percent of its Original with Increasing Update Interval</td>
<td>89</td>
</tr>
<tr>
<td>Figure 7.8</td>
<td>Flow Distribution in Each Source During 15 Seconds</td>
<td>90</td>
</tr>
<tr>
<td>Figure 7.9</td>
<td>Packet Drops with Increasing Flows for Each Source-Destination Pair when Update Interval is 9 Seconds</td>
<td>91</td>
</tr>
<tr>
<td>Figure 7.10</td>
<td>Packet in Queue when Each Node Triggers 7 Sessions of Flow and Update Interval is 9 Seconds</td>
<td>92</td>
</tr>
<tr>
<td>Figure 7.11</td>
<td>Packet in Queue when Each Node Triggers 8 Sessions of Flow and Update Interval is 9 Seconds</td>
<td>92</td>
</tr>
<tr>
<td>Figure 7.12</td>
<td>Packet Drops with Increasing Flows in Each Source-Destination Pair when Update is Based on Link Utilization</td>
<td>93</td>
</tr>
<tr>
<td>Figure 7.13</td>
<td>Updates with Increasing Flows in Each Source-Destination Pair when Update is Based on Link Utilization</td>
<td>93</td>
</tr>
<tr>
<td>Figure 7.14</td>
<td>Packet Drops with Increasing Flows in Each Source-Destination Pair in Evenly Loaded and Quasi-Static Situation</td>
<td>96</td>
</tr>
<tr>
<td>Figure 7.15</td>
<td>Packet Drops with Increasing Flows in Each Heavily Loaded Source-Destination Pair in Unevenly Loaded Situation</td>
<td>98</td>
</tr>
<tr>
<td>Figure 7.16</td>
<td>Packet Drops with Increasing Flows in Each Heavily Loaded Source-Destination Pair in Heavily Unevenly Loaded Situation</td>
<td>99</td>
</tr>
<tr>
<td>Figure 7.17</td>
<td>MCI Topology</td>
<td>101</td>
</tr>
<tr>
<td>Figure 7.18</td>
<td>MCI Topology Used in the Simulation</td>
<td>102</td>
</tr>
<tr>
<td>Figure 7.19</td>
<td>Packet Drops with Increasing Heavier Sessions in Unevenly Loaded Situation</td>
<td>103</td>
</tr>
<tr>
<td>Figure 7.20</td>
<td>Packet Drops with Bucket Sampling Interval in Dynamic</td>
<td>104</td>
</tr>
</tbody>
</table>

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 7.21 Call Blocks with Increasing Flows in Each Heavily Loaded Source-Destination Pair in Unevenly Loaded Situation .......... 105
Figure 7.22 Call Blocks with Increasing Call Number in Unevenly loaded Situation ...................................................... 106
ACKNOWLEDGMENTS

I would like to express my sincere thanks to my advisor, Dr. Shahram Latifi, for his guidance, encouragement, support and friendship. He has contributed greatly to this dissertation and my maturity as a researcher.

I wish to thank the members of my dissertation committee: Lori M. Bruce, Eugene McGaugh, Ajoy K. Datta and Kazem Taghva. Dr. Bruce took time to share her research experience with me. I enjoyed our friendly discussions, both technical and otherwise. Dr. Datta reviewed many chapters of the dissertation and gave me suggestions to refine many of the ideas presented here. Dr. McGaugh and Dr. Taghva have always expressed their concern for this work each time we met.

Several people have assisted in this research. I would like to acknowledge Jay Nietling and Steven E. Lumos for their help on installing ns network simulator. Jay has answered my many questions about Unix systems. Dr. Srihari Nelakudit (UMN) sent me his ATM/PNNI simulation code, which helped me to understand and enhance ns for my simulation. Prof. Lixia Zhang (UCLA) gave me valuable suggestions on my research direction and provoked my interest on IETF.

I greatly appreciate the financial support of UNLV National Super-computing Center for Energy and Environment and the Graduate College.

Finally I am especially thankful for the caring and support of my mother and my in-laws. I would like to thank my husband, Chenyong, for the love, support, and encouragement which made it possible for me to overcome the obstacles in the life of a graduate student. He has put up with my fears and frustrations. I would also like to thank my son, Junxiong, for his understanding; I owe him so many hours of play and attention.
CHAPTER 1

INTRODUCTION

The new generation of network services is being developed for incorporation in communication infrastructure. These services, generally called Quality of Services (QoS), should accommodate data file, video, and audio applications. The Internet is a global communications network, which is used by tens of millions of people in the world for business, education and recreation. To make the Internet serve multiple applications with different, and in many cases, conflicting requirements is a great challenge for researchers.

The Internet, whose standard suite is TCP/IP, was intended to transfer data application; therefore, its communication is connectionless. To support connection-oriented traffic (video or audio) with stringent requirements for bandwidth and delay assurance, new protocols must be added to the suite. The research is under way to make Internet-style packet-switched networks capable of supporting real-time applications.

Special-purpose computers called routers connect the Internet sites together. As data is forwarded from one place in the Internet to another, it is the routers that make the decisions as to where and how the data is forwarded. Routing is a mechanism to keep the Internet running smoothly. Although many users of the Internet and the World Wide Web are unaware of the machinery underlying the network applications, routing is an interesting but complicated subject. Routing protocols are sophisticated distributed algorithms that must also be extremely robust to protect a large, decentralized network like the Internet from being out of service.
The new requirements from diverse applications further increase the complexity of the Internet routing. This dissertation tackles the limitations of the current Internet routing architecture and proposes a new multipath routing method to improve Internet routing for supporting new services requirements.

Multipath routing has been proposed to balance network load. Since real-time traffic cannot change its packet rate as flexibly as data traffic does, multipath routing becomes more important in the real-time environment. On the other hand, in response to current rapid growth in the size of the Internet and demand for network bandwidth, some networks have been designed with much more multiple paths [21]. Therefore, the research on multipath routing is becoming more meaningful and practical than ever.

Current Internet routing utilizes the simple traffic forwarding method, which splits the traffic load equally among multiple paths. For data traffic, the traffic forwarding method works well as long as the difference of delay on the multiple paths is not too big. For real-time traffic, however, the delay difference produces delay variation and may degrade the performance at the application layer. How to efficiently utilize the multipath to transfer real-time traffic is an important issue; however, there are few reports addressing this problem. To overcome the inefficiency of current multipath routing techniques, a new multipath routing architecture is proposed in this dissertation. This dissertation addresses the requirements of routing in multimedia environment with a multipath routing scheme, referred to as Two-level Class-based Multipath Routing with Prediction (TCMP). The key features of the TCMP method are the use of a long-term first-level routing metric and a short-term second-level routing metric to construct routing tables. Furthermore, the leaky bucket scheduling mechanism is adopted. Instead of regulating traffic rate, the proposed routing scheme uses leaky bucket to monitor the bottleneck bandwidth on each path during the net-
work stable period, and to guide traffic forwarding. This routing scheme is analyzed theoretically, and compared to other routing schemes that could be used in future Internet. Simulation has proved that the TCMP not only decreases the end-to-end delay and increases the amount of traffic a network can carry, but also avoids high routing overhead and eliminates network oscillation.

Chapter 2 introduces the current Internet routing architecture and presents its limitations to support real-time and non-real-time applications. Chapter 3 surveys related work in the area of network routing, and discusses its applicability to the specific problems of QoS-based routing in the Internet. Chapter 4 presents some considerations about multiple metrics and multiple paths routing. In Chapter 5, the two-level routing architecture, e.g. TCMP is presented. The worst case end-to-end delay is derived in Chapter 5. The performance of TCMP routing algorithm is examined by simulation in Chapter 6. Chapter 7 provides a summary of the results and contributions, and discusses areas worthy of future investigation.
CHAPTER 2

INTERNET ROUTING ARCHITECTURE AND LIMITATIONS

This chapter introduces the Internet Routing Architecture, and some basic routing characteristics. After analyzing the requirements for real-time applications, we present the limitations of current Internet routing architecture for supporting real-time traffic. This is followed by a discussion on the design goals of QoS-based Routing.

2.1 Internet Routing Architecture Overview

The Internet is organized into regions called Autonomous Systems (ASs). Each AS consists of a collection of routers under the control of a single administrative entity. For example, all the routers in an AS belong to a particular Internet Service Provider (ISP), corporation, or university. The collection of ASs is organized in a rough hierarchical fashion. The core of the Internet is on the top of the hierarchy. The closer to the core of the Internet, the more routes appear in the AS. The ASs at the core of the Internet carry the complete routing table, currently including 45,000 routes, and do not use a default route (they are in the so-called default-free zone) [65]. All other ASs use a default route, pointing up the hierarchy, enabling them to carry only a subset of the Internet's routes. This arrangement of ASs is pictured in Figure 2.1. If the two providers are at the same level of the hierarchy, there will be a simple agreement to exchange routing information. However, when one AS is lower in the hierarchy (downstream), this AS is sometimes entering into a customer relationship with the upstream provider. This means that the upstream provider will
advertise the downstream’s addresses to the rest of the Internet, and will forward the downstream’s packets to other providers and their customers as appropriate. In other words, the upstream provider provides transit for the downstream provider.

The identity of the ASs at the Internet’s core has changed over the years. Originally the ARPANET network was at the Internet’s core. Then in 1985, the National Science Foundation funded a new Internet core, called the NSFNET. In 1987, the NSFNET was upgraded to be interconnected with T1 lines, and in 1992, the line speed was upgraded to T3. The NSFNET was decommissioned in 1995. Today the Internet’s core consists of around half a dozen commercial Internet providers, including UUNET, MCI, and Sprint. Dozens of the providers may connect at a single exchange point. Physically these exchanges are usually implemented as bridged FDDI/Ethernet combinations or as ATM subnets. Two providers may also directly communicate over a private connection, such as a high-speed leased line or an ATM.
circuit. This kind of private peering is becoming common between the top-level ISPs making up the Internet core.

The routers within an AS exchange routing information via a common routing protocol, for example OSPF (Open Shortest Path First) [64], whereas a different routing protocol was used to exchange routing information between ASs, such as BGP (Border Gateway Protocol)[77]. This dissertation focuses on multipath routing in AS. The proposed multipath routing scheme can be treated as an extension of the OSPF protocol. To give an insight into routing mechanisms, the following section presents the characteristics of the routing algorithms.

2.2 Routing Characteristics

According to the OSI seven-layer model, routing is a network layer function that determines the paths from source to destination for traffic flows. The times at which routing decisions are made depend on whether the network uses datagrams or virtual circuits. In a datagram network, such as the Internet, a routing decision is made for every incoming packet and the route to a destination can be changed at any time. In a virtual circuit network, such as ATM (Asynchronous Transfer Mode) network, routing decisions are made when a new virtual circuit is being set up. All data packets subsequently follow the established route until the session is terminated or reset.

The functions of a routing algorithm are at two levels. At the basic level, a routing algorithm has to maintain the reachability of the network. When parts of the network fail, a routing algorithm has to find alternative paths when they do exist. At a higher level, a routing algorithm has to ensure optimal and fair sharing of the network so that resources are efficiently utilized. The difficulty in routing is due to the distributed nature of the operation. A routing algorithm has to deal with resource failures and traffic changes with incomplete and delayed information feedback.
Routing as a complex decision making procedure has many different but related functions. A routing algorithm has to monitor the network status and collect information which routing decisions can be based on. The collected information should then be propagated over the network in a timely fashion. The routing table can then be produced for all destinations in the network, and finally it has to forward packets to the next hop along the route. In the following sections, the four functions of the routing algorithm, namely distance estimation, route computation, information propagation, and packet forwarding, are discussed.

2.2.1 Distance Estimation

A routing algorithm has to make routing decisions based on the current state of the network. It has to continuously monitor and collect information to maintain the database up-to-date. One node may collect information about the network state by (1) measuring local information to which the node has direct access, e.g. output queue length, link utilization; (2) receiving updates from other nodes which contain explicit remote information such as delay, queue length; (3) learning implicitly from the packets it receives from other nodes.

The frequency at which the information is updated is important. Highly frequent updating may improve the accuracy, but it may also cause a substantial amount of overhead. The route updating period has to be decided according to the network environment.

2.2.2 Route Computation

The process of route computation is the heart of the routing algorithm. It determines the best routes for traffic through the network based on the information collected so far. The shortest-path algorithms have been widely used in route com-
putation in which the routing algorithm attempts to optimize the performance by minimizing the distance of the route. There are two main groups of shortest-path algorithms: distance-vector algorithms and link-state algorithms.

**Distance-Vector Algorithms**

In a distance-vector algorithm, each node maintains a routing table containing the distance of the shortest path to every destination in the network. A node only informs its immediate neighbors of any distance changes to any particular destinations. The distance-vector algorithms are based on an algorithm developed by Ford-Bellman. The idea is to compute the shortest paths from every node to every other node by repeating a distributed version of Ford-Bellman algorithm [12].

Let $D_i^{(h)}$ be the shortest path length from source node 1 to node $i$, subject to the constraint that the path contains at most $h$ arcs. We take $D_i^{(h)} = 0$ for all $h$. Let $d_{ij}$ be the length of path between the adjacent node $i$ and node $j$ and $d_{ij} = \infty$ if the $(i,j)$ is not an arc of the graph. The Ford-Bellman algorithm can be written as:

Initially, $D_i^{(0)} = \infty$, for all $i \neq 1$
For each successive $h > 0$, $D_i^{(h+1)} = \min[D_j^{(h)} + d_{ji}], i \neq 1$

In the distance-vector algorithm, nodes do not have complete topology information. When link distance changes, the algorithm has to update the routing table by recomputing the shortest paths over the entire network. Before the computation is completed, the routing table may not be consistent and loops may be formed. Examples of distance-vector algorithms include the old ARPANET routing algorithm [62], Cisco's EIGRP [8] and BGPv4 [77].

**Link-State Algorithms**

In a link-state algorithm, each node keeps track of the entire network topology and computes the routing table based on the link distance information broadcast by every
node in the network. Routing loops may exist during the updating period but routing tables eventually become consistent when each node has updated its routing table. Link-state algorithms have been used in the OSPF [64, 65], ATM/PNNI [74] and IS-IS [73].

The shortest path algorithm used in link-state algorithms is one developed by Dijkstra. Dijkstra algorithm belongs to label setting method. The basic idea is to find the shortest paths in order of increasing path length. We view each node $i$ as being labeled with an estimate $D_i$ of the shortest path length to a specific node 1. When the estimate becomes certain, we regard the node as being permanently labeled and keep track of this with a set $P$ of permanently labeled nodes. The node added to $P$ at each step will be the closest to node 1 out of those that are not yet in $P$. The algorithm is as follows:

Initially $P = 1, D_1 = 0$, and $D_j = d_{ij}$ for $j \neq 1$.

Step 1: finding the next closest node.

Find $i \notin P$ such that $D_i = \min_{j \notin P} D_j$, $j \notin P$

$P = P \cup \{i\}$.

If $P$ contains all nodes then terminate. Otherwise

Step 2: updating the labels.

For all $j \notin P$, $D_j = \min[D_j, D_i + d_{ij}]$

Goto Step 1.

Routing algorithms based on the Dijkstra algorithm often use flooding to propagate information, which is fast and robust.

The routing decisions can be made in many different ways. In a fixed routing algorithm, the computation might be done off-line and fixed for a relatively long time. On the other hand, an adaptive routing algorithm may update its routing table whenever significant changes are detected. In an adaptive routing algorithm, the
route computation can be carried out in a centralized or distributed fashion.

**Centralized Routing**

In a centralized routing algorithm, also called source routing algorithm, routing decisions are made only in one or a few centers and distributed to each node in the network; therefore, the overhead is reduced on other nodes and sophisticated algorithms such as disjoint multi-path algorithm can be used. The examples for using centralized routing algorithms are ATM/PNNI routing protocol [74], or Policy routing [84], etc. There are also many disadvantages in centralized routing. One serious problem is that any failure of the centers may lead to catastrophic results. In a large network, the computation may take an unacceptable time even on a high performance CPU, and the route updates can also consume large amount of bandwidth. The routing traffic is heavily concentrated on the lines leading to the centers. The resultant heavy load and possible congestion make the centers more vulnerable.

**Distributed Routing**

In a distributed routing algorithm, which is also called hop-by-hop routing, all nodes participate in the process of decision making. Most modern routing algorithms, such as OSPF and BGP etc., fall into this class. The most important advantage of distributed routing is its high survivability in the face of link or node failures. Distributed computation also reduces the amount of information that has to propagate. However, distributed routing computations are usually more complex. The routing algorithms must ensure that the distributed status information and routing tables are consistent among all nodes, otherwise long-lasting routing loops may form which may have severe effects on routing performance.
2.2.3 Information Propagation

Information about changes in network topology and traffic load has to be propagated to other nodes so that adjustments to routing tables can be made. The procedure of information propagation must meet high efficiency and high reliability criteria. In the Internet intra-domain protocol OSPF, flooding protocol is used to disseminate network information.

2.2.4 Packet Forwarding

The route computation results in a routing table. In many single path routing algorithms, the node looks up the destination in the routing table to obtain the number of the output lines and forwards the packet to the next hop. In some algorithms, there may be more than one route for a destination. The nodes have to select one route based on some pre-specified criteria. For example, in a multipath routing algorithm, packets are forwarded to several output lines according to certain probabilities.

2.3 Requirements for New Services

One of the most significant performance complaints of real users today is that large data transfers take too long, and that there is no way to adjust or correct for this situation. People who would pay more for a better service cannot do so, because the Internet contains no mechanism to enhance their service. Traditionally, network providers have tended to provide all of their customers with the same type of performance. Traffic is processed as quickly as possible, but there is no guarantee as to timeliness or actual delivery. It is becoming apparent the several service classes will likely be demanded for the future Internet. In addition to the Best Effort service, the new services include:

- Premium Service will provide low delay and low delay jitter (delay variation)
to real-time applications such as Internet Telephony and Video Conferencing. This service is a fundamentally different from the Best Effort service.

- Assured Service will provide reliable and predictable services to some applications that need reliable transmission, even in time of network congestion. For example, some companies that do business on the Web, will be willing to pay a certain price to make their services reliable and to give their users a fast feel of their Web sites.

In the following sections, we illustrate above new services.

2.3.1 Premium Service

A Premium Service traffic flow generally needs real-time transmission. The first well-known real-time application in the Internet was the audio conference of IETF (Internet Engineering Task Force) in 1992, which was served over the IP multicast backbone (MBone) [32]. From then, the MBone has served as a testbed for the development of multicast protocols and group conferencing tools, such as well-known conferencing tools vat for audio and nv for video [61]. These applications are very useful, but the quality of the audio and video received varies over time and location from good to very poor, depending on the network conditions. The main reason for the poor performance is the nature of the real-time traffic.

Real-time applications are quite different from standard data applications, and require services that cannot be delivered within the typical data service architecture, such as the Internet. Clark et al. gave a very detailed analysis of the properties of the real-time traffic [27]. Wang also analyzed burstiness and jitter in multimedia communications [89, 90]. The vast majority of future real-time applications including most video and audio applications, such as Internet telephony, video conferencing, will be fit to a particular class of real-time applications called play—back applications. In a
play-back application, the source takes some signal, packetizes it, and then transmits it over the network. The network inevitably introduces some variation in the delay of each delivered packet. This variation has traditionally been called jitter. The receiver depacketizes the data and then attempts to faithfully play back the signal. This is done by buffering the incoming data to remove the network induced jitter and then replaying the signal at some designated play-back point. Any data that arrives before its associated play-back point can be used to reconstruct the signal; data arriving after the play-back point is useless in reconstructing the real-time signal.

Not all real-time applications are play-back applications, for example, a visualization application which merely displayed the image encoded in each packet whenever it arrived.

Play-back real-time applications have several service requirements which are explained below:

1. Since there is often real-time interaction between the two ends of an application, as in a voice conversation, the application performance is sensitive to data delivery delay; in general lower delay is much preferable.

2. To set the play-back point, the application needs to have some information about the delays that each packet will experience.

3. Applications with real-time interaction is also sensitive to delay jitter. Delay variation is generally more critical than delay as long as the delay is not too high. For example, when listening to a speech or a concert, the delays are typically less important than fidelity.

4. These play-back applications can often tolerate the loss of a certain fraction of packets with only a minimal distortion in the signal.
Therefore, real-time applications generally require performance guarantee from the network, in terms of the bandwidth received, delay encountered, or packet loss rate experienced. Comparing the above play-back real-time application with traditional data applications (electronic mail, file transfer, and remote login), the different requirements between these two types of traffic can be easily recognized. The performance of traditional data applications is largely dominated by the average delay that the packets have experienced. The delay variation or jitter often has little impact on these applications. Data applications are usually called elastic applications or non-real-time applications, because they can tolerate considerable delay. Of course, these applications have specific requirements. We introduce the requirements in the next section.

2.3.2 Assured Service

Similar to Best Effort service, the Assured Service does not have quantifiable timing requirements (delay or delay variation), but it assures that the user’s traffic is unlikely to be dropped as long as it stays within the expected capacity profile (transmission rate).

One should note that the traffic on the Internet is a mixed of data objects from different users, with different sizes and different objectives as to overall delivery time. One user may be transferring a single keystroke, with the goal of delivery in a fraction of second. Another user may be transferring an image of many megabytes, with the goal of delivery within five minutes. A third user may be connecting to a succession of locations across the Internet, and transferring an unpredictable number of bytes from each before moving on. In general, the faster a packet network delivers a data object, the greater the user satisfaction. However, as illustrated in the paper [28], packet delay is not an indication of service quality for Assured Services, throughput
or packet drops probability is the criterion that the user has for evaluating network performance.

Currently four classes with three levels of drop precedence in each class are defined for general use [44]. The three levels are low drop, medium drop and high drop precedences. The classes are Class 1, Class 2, Class 3, and Class 4. As an example, Assured Service could be used to implement the so-called Olympic Service, which consists of three service classes: gold, silver, and bronze. These three service classes can be mapped to the Class 1, Class 2 and Class 3. Packets are assigned to these three classes so that packets in the gold class have greater probability for timely forwarding than packets assigned to the silver class. The same kind of relationship exists between the silver class and the bronze class. If desired, packets within each class may be further separated by giving them either low, medium, or high drop precedence.

Today’s Internet does not have a mechanism to provide Premium Service and Assured Services. The stringent performance requirements of real-time applications and requirements of offer different levels of forwarding assurances for other applications necessitate a re-examination of the fundamental architectural components of today’s Internet. Network routing, which determines the sequence of links a packet traverses between source and destination, is one such component. In this dissertation, we address the Internet routing issues for providing Premium Service and Assured Service. The following section examines the limitations of routing architecture in the Internet.

2.4 The Limitations of Current Internet Routing Architecture

It is noted that current Internet routing architecture is inadequate for real-time applications, which often require guaranteed quality of service. There are many papers that analyze the limitations of today’s routing architecture and propose the
appropriate solutions (see [24, 25, 29, 58, 89, 90, 91] as examples). The limitations are summarized in the following sections.

2.4.1 Single Metric

A routing metric is an attribute of a path that consist of the cost from a source to a destination. A metric can be a link cost, link delay and bandwidth, etc. The original ARPANET routing architecture used distance-vector algorithm based on Ford-Bellman algorithm. However, to solve the problem of routing overhead and convergence, the routing algorithm was replaced by a link-state algorithm which is also called SPF (Shortest Path First) algorithm due to Dijkstra [12]. The routing overhead includes state distribution, state storage and route calculation.

The link metric was based on queueing delay measured by each node. The routing tables were calculated based on delay change in every 128 milliseconds. Running result of the network showed that routing based on link delay could not perform well at high load when queueing delay was a significant part of measured link delay, which consists of queueing, transmission, and propagation delays. This is mainly because of the classical delay-utilization curve; a small increase in utilization corresponds to a large increase in link delay. This dramatic change can result in the link becoming unattractive and thus being avoided by all delay-sensitive sources. Consequently, at the next routing update the link reports a very low cost and can become attractive again. This leads to oscillatory behavior, which in turn degrades performance [51]. In the current Internet routing protocol OSPF, link metric is based on hop-count or administrative weight.

The IP layer of the Internet Protocol suite specifies different TOS (Type of Service) [73]. Among them are the minimum delay service required for example by interactive traffic or real-time traffic (e.g. audio), and the maximum throughput service required.
for example by bulk transfers such as network mail or FTP. Routing protocols such as
the Internet OSPF and the OSI IS-IS [73] provide separate next-hops for each TOS.
However, the TOS mechanism has been so far of little use, and little is known on how
well it would work in practice.

2.4.2 Single Path Routing

The current Internet routing protocol uses single path routing algorithm. The
fundamental problem for the single path routing is uneven traffic distribution. The
reason is that the shortest path is always selected to forward packets. As a result,
routers and links along the shortest path between two nodes may become congested
while routers and links along a longer path are idle. In this situation, network con­
gestion and oscillation can occur. Network congestion means that all routers and
links along a path are overloaded. For the Internet routing architecture, although
OSPF does allow a router to alternate among several equal cost paths to a destina­
tion, alternate paths with acceptable but non-optimal cost can not be used to route
traffic [65]. On the other hand, the Equal Cost Multipath (ECMP) [45] option of
OSPF is useful in distributing load to several shortest paths. However, packets for
the same traffic may experience different end-to-end delay because of being transfered
to different path among equal paths.

2.4.3 Single Service

The Internet was derived from ARPANET [62] to service data transmission, so
the Internet has only one service, called Best Effort service. This service means the
network accepts all flows from users and tries to transmit as much as it can, based
on a FIFO (First In First Out) scheduling algorithm. There is no admission control
and the network offers no assurance about when, or even if, packets will be delivered.
There are two types of transport protocol on the Internet; these protocols are TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). TCP provides a connection-oriented, reliable, full-duplex, byte-stream delivery service to a data application, such as Telnet and FTP. UDP is a simplistic protocol that does not provide for congestion management, packet loss notification feedback, or error correction. UDP assumes these will be handled by a higher-layer protocol. SNMP (Simple Network Management Protocol) is reliant on UDP. Figure 2.2 shows these two types of function.

TCP can adjust the transmission rate of an application. For the Best Effort service, TCP works very well in practice. However, for Assured service described in previous section, TCP cannot fulfill its requirement. The reason is that TCP cyclically increases its sending rate. It will just send faster if it discovers unused bandwidth. As a result, one user may get satisfactory service and others do not. Future Internet must combine various uses in a way that makes each of the users sufficiently satisfied.

For the Premium Service, since real-time traffic cannot tolerate the acknowledgment delay, and it also does not need high reliability, currently the UDP protocol is used to transmit real-time applications.
Real-time applications do not perform adequately when running over the current Internet because the variations in delay are too extreme and there are too many dropped packets. The real-time applications typically do not back off in the presence of congestion by using UDP as transmission service; they have a consistent transmission rate. On the other hand, when the real-time applications are contending for bandwidth with traditional data applications, since data applications can tune their transmission rate when the TCP is used as service function, they end up receiving very little bandwidth. Thus, when running in the current Internet, real-time applications do not always perform adequately; they also often interfere with data applications. The following example illustrates the problem.

**Example:** Consider a single link network with the exponential departure rate $\mu = 10$ units/sec and the server uses a FIFO service. There are two network clients with Poisson arrival rate $r_1 = r_2 = 4$ units/sec respectively. The utility function $U_i$ is defined to describe how the performance of an application $i$ depends on the experienced delay; increasing $U_i$ reflects increasing application performance [82]. Now let $U_1 = 1 - 2d_1$ and $U_2 = 1 - d_2$, where $d_1$, $d_2$ represent the average queueing delay delivered to client 1 and client 2 respectively. It is clear that client 1 is more sensitive to queueing delay. Assume a M/M/1 model for the network queue. The average delay is $d = 1/(\mu - r)$, thus, $d_1 = d_2 = 1/(10 - 4 - 4) = 0.5$ and $U_1 = 0, U_2 = 0.5$. To improve application performance, two situations can be considered:

1. First, assuming that client 1 and 2 are all data traffic, so these two clients can tune their arrival rates. For example,
   - $r_1 = 3$ and $r_2 = 3$, so, $d_1 = d_2 = 1/(10 - 3 - 3) = 0.25$ and $U_1 = 0.5, U_2 = 0.75$. 

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
2. Second, assuming client 1 is real-time traffic. So its arrival rate is constant (Some papers discuss some real-time applications having variable transmission rate [27, 82]. Here traditional real-time traffic is considered. Even though real-time applications can have variable arrival rate, they can not work in the same way as non-real-time traffic can). To obtain the same utilization function as the above example at the first time, \( r_1 = 4 \) and \( r_2 \) has to be equal to 2, so \( d_1 = d_2 = 1/(10 - 4 - 2) = 0.25 \). Furthermore, to get the same utilization function as above, e.g. \( d_1 = d_2 = 0.2 \), and \( r_1 = 4, r_2 \) has to be equal to 1. Thus, it can be seen that the real-time traffic not only affects non-real-time traffic, but also obtains degraded performance some time. For example, when \( r_1 = r_2 = 4, U_1 = 0 \).

One can address this problem by modifying the application implementations rather than the network implementation, such as \( nv \) and \( vat \) [61]. The method is up to applications to adjust to the inevitable variations in packet delay and available bandwidth. There are likely to be limitations to this adaptability. Otherwise, one can address these problems without changing the basic Internet architecture by improving different aspects of the implementations. For example, many works focus on using scheduling algorithms such as: Fair Queueing algorithms [80], priority queueing algorithms [27] in routers. We address how to enhance the Internet routing architecture to support real-time applications.
2.5 Design Goals of QoS-based Routing

This section first presents the general design goals for routing algorithms; then based on these goals, the added design requirements for real-time traffic are analyzed.

2.5.1 General Design Goals for Routing Algorithms

Since routing is an important mechanism in the Internet, an efficient and reliable routing algorithm is essential to make the Internet run smoothly. On the other hand, stability and adaptability are also the basic requirements for routing.

**Efficiency**

The operation of routing consumes network resources such as CPU resources and link bandwidth. It is important that routing algorithms are simple and efficient so that the processing and transmission of normal data packets are not affected. The efficiency can be measured in terms of computational complexity, storage complexity, and communication complexity. In some cases, precise measurement is difficult to obtain and the worst-case performance may be used as an indicator. A tradeoff has to be made between the functionality and the overhead.

**Reliability**

The routing algorithm is one of the critical components in the network. Its reliability and robustness are of vital importance. The behavior of a routing algorithm must be predictable. It is desirable that a routing algorithm have the ability to carry out consistency checks and eliminate suspicious routing updates so that it may survive both malfunctions and deliberate attacks.
Stability

Since routing is a distributed operation, stability is important. Inconsistent routing information or poor route computation can cause routing loops and generate large amounts of artificial traffic, which in some cases can bring down the network. For a given topology and traffic conditions, the routing algorithm should eventually converge to a steady state free of routing loops and route oscillation.

Adaptability

One of the basic functions of routing is to deal with topological changes and maintain reachability. When topology changes as a result of failures and repairs, a routing algorithm has to be able to rebuild the routing table automatically. The ability to respond to topological changes depends on the information exchange. A tradeoff must be made between the speed of adaptive action and the routing overhead. Routing algorithms can not change faster than relevant information can be propagated to the decision point.

Optimality

The ultimate goal of a routing algorithm is to achieve optimal resource sharing. The quality of a routing algorithm is determined by both the satisfaction of individual users and the efficiency of the network resource utilization. A routing algorithm should produce routes that meet the individual requirements of the users and in the mean time take into account the global requirements of the network.

2.5.2 Design Goals for Routing to Support New Services

QoS-based routing computes paths having available resources to satisfy application performance requirements. For a network to support QoS requirements, routing must supply explicit information on resources available in the network so that packets
of various applications can be routed on proper paths based on QoS requirements of these applications. The objectives of QoS-based routing can be defined as follows:

1. Dynamic determination of feasible paths: QoS-based routing can determine a path, from among possibly many choices that has a good chance of accommodating the QoS of the given flow.

2. Constructing routing table based on the link state, QoS-based routing scheme can aid in the efficient utilization of network resources by improving the total network throughput. Such a routing scheme can be the basis for efficient network engineering.

3. Efficiently limiting routing overhead when QoS routing implements a dynamical changing of routing table according to network load.

Given the objectives of QoS-based routing, the question arises: what routing metrics are used and how are QoS-accommodating paths computed for unicast flow? What is the granularity of routing decision (i.e. destination-based, source and destination-based, or flow-based)? What factors affect the routing overheads? And how is scalability achieved? These are the questions addressed in the dissertation. In the next chapter, the related work is introduced and its applicability is discussed.
CHAPTER 3

RELATED WORK AND MOTIVATION

Many studies in the literature have addressed multipath routing and different aspects of QoS-based routing problems. This chapter gives a brief survey of related work, which includes the following aspects:

1. We survey Constraint-Based Routing (CBR) problems. Because of the high demand for the Internet to transfer multimedia applications, CBR has recently attracted more attentions.

2. Multipath routing has been studied in the rich literature on network routing to solve network congestion and load balancing problems. We introduce multipath routing for congestion control and load balancing, and further introduce current work on multipath routing for transferring multimedia traffic.

3. Research on dynamic routing in circuit-switched network has a long history. Many of the concepts found in circuit-switched routing can be applied to QoS-based routing. Although packet-switched networks are much different from circuit-switched networks, it is instructive to review dynamic routing methodologies associated with circuit-switched network. This helps one understand the routing problems encountered, and provide possible solutions for packet-switched routing.

4. Work on routing in High Speed Networks is also discussed. Similar to routing in circuit-switched networks, routing in High Speed Networks can also lead to
solving packet-switched routing problems.

5. Most current work in real-time applications has concentrated on specifying packet scheduling algorithms, flow specifications, admission control algorithms and reservation protocols. Understanding this work is very important for providing an efficient routing algorithm to increase network throughput and reduce end-to-end delay.

6. Traffic engineering is the most important aspect in networks where multiple parallel or alternate paths are available. Since we tackle multipath routing problems, traffic engineering is the main issue we discuss.

3.1 Constraint-based Routing

Many parameters can be used to characterize network resources, such as bandwidth, loss probability, delay, delay jitter, cost, etc.. However, it may not be feasible to have these parameters as metrics, since the problem of finding a path that is subject to multiple constraints is inherently difficult. Constraint-Based Routing (CBR) is used to compute routes that are subject to these multiple constraints. The CBR evolves from QoS-based routing. Given the QoS request of a flow or an aggregation of flows, QoS-based routing returns the route that is most likely to meet the QoS requirements.

The CBR is widely studied for supporting multimedia applications. Routing algorithms are expected to satisfy certain additional constraints to make them suitable for actual practical implementation on wide area networks. A well-known theorem in Constraint-Based Routing is that computing optimal routes subject to constraints of two or more of parameters, such as loss probability, delay, delay jitter, cost, are NP-complete. The theorem is based on the assumptions that all metrics are independent [89]. Feasible combinations of metrics should only contain the bandwidth and one of
parameters listed above.

Although the assumption in the above NP-complete problem may be true in a circuit-switched network, the bandwidth, delay or delay jitter are not independent in packet networks. As a result, polynomial algorithms for computing routes with hop-count, delay, and jitter constraints exist [56]. The complexity of such algorithms is \( O(N \cdot E \cdot e) \), where \( N \) is the hop-count, \( E \) is the number of links of the network, and \( e \leq E \) is the number of distinct bandwidth values among all links. Nevertheless, the theorem can qualitatively present the complexity of a routing algorithm: a complex algorithm in circuit-switched networks is still complex in packet networks, although it may not be NP-complete.

Jaffe [47] studied a variation of the problem, in which both cost and delay were specified as constraints, and proposed pseudo-polynomial-time and polynomial-time heuristics for solving the problem. Sriram et al. [83] adapted the preferred link routing approach to delay-constrained least-cost routing for real-time channel establishment. They presented a set of heuristic functions which mainly used local information to make route selection decisions, so that the algorithms were suitable for wide area networks.

The shortest-widest routing algorithm has been employed as a mechanism for QoS routing, where a shortest-widest path is a path with the shortest propagation delay among all paths with the largest bottleneck bandwidth from source to destination. The algorithm is to find a path with maximum bottleneck bandwidth (the widest path); and when there are more than one widest path, the one with the shortest propagation delay is chosen [89].

Since OSPF has worked well for routing data applications, the natural way to improve Internet routing performance is to extend OSPF to build a routing table using more routing metrics, such as bandwidth and delay. An extended OSPF method
was proposed by Guerin et. al. [41]. In this draft, the metrics on which the path selection process is based are: link available bandwidth which can be quantized to reduce overhead, hop-count and policy. The path selection algorithm picks a path with the minimum possible number of hops among those that can support the requested bandwidth. When several such paths are available, the preference is for the path whose available bandwidth is maximal. By using policy routing, long propagation delay paths, such as satellite links, are eliminated before doing path selection. Another OSPF extension is called QOSPF proposed by Zhang et. al. [96]. In QOSPF, a router calculates a routing table using network topology, network resource information, and QoS requirements for the flow. Routing for QoS flows is based on (source, destination), and routing computations are triggered by external events. The initial trigger for QoS routing computation comes from a resource reservation protocol such as an RSVP Path message [17]. QOSPF determines QoS routes based on source and destination addresses. This implies that all traffic between a given source and destination, regardless of the flow, will travel down the same route. Again, the route must have capacity for all the QoS traffic for the source/destination pair. The amount of routing state also increases since the routing tables must include source/destination pairs instead of just the destination.

The best granularity is found when routing is based on individual flows. In paper [40], a flow-based routing mechanism is proposed. The mechanism provides resources reserved by a flow for hop-by-hop routing. Each QoS flow can be routed separately between any source and destination. However, flow-based routing incurs a tremendous cost in terms of the routing state.

Since the numbers of traffic flows getting into the Internet are generally very high, there may be always more than one traffic flow getting into the same router within the stable period of network. Single path routing algorithms proposed for supporting
multimedia traffic have their limitations. For example, for the shortest-widest routing scheme, if traffic flows can utilize the multiple equal bandwidth paths, the problem of overloading on the shortest-widest path can be avoided. The following section surveys the multipath routing algorithms.

3.2 Multipath Routing in Packet-Switched Networks

Multipath routing algorithms are proposed in current work for transferring multimedia traffic. Villamizar proposed an optimized multi-path to extend OSPF. When using the optimized multi-path routing method, loading information is flooded within an OSPF area and traffic is split according to loading levels on each path [88]. Matta proposed to classify traffic as delay-sensitive and throughput-sensitive and route these two types of traffic using low delay routes and under-utilized routes, respectively [58]. Type-of-service queueing is also used to isolate the two traffic classes. In [75], Rao and Batsell showed two NP-complete problems. One problem is finding multipath to transmit traffic at a bounded end-to-end delay; another problem is finding multipath to transmit traffic within a limited delay jitter. Rao and Batsell proposed a multipath routing algorithm to satisfy end-to-end delay requirement. The application has to be split according to the rules they derived and then the split traffics are routed to different paths.

Source routing, also called explicit routing is a very powerful technique which potentially can be useful for a variety of purposes. However, with pure datagram routing, the overhead of carrying a completely explicit routing with each packet is prohibitive. Breslau proposed an adaptive source routing to support real-time applications [18]. The main reason that he proposed the source routing scheme is to prevent routing loops when an alternate routing architecture is developed. Alternate routing means that a node uses a route that has higher cost than the minimum
cost route. Generally, routing loops are caused by using hop-by-hop routing scheme when nodes make alternate routing decisions in an uncoordinated or unconstrained manner. We illustrate the routing loop problem in the following example.

Figure 3.1 shows a six-node network. The shortest path from $n_1$ to $n_6$ is via nodes $n_2$ and $n_5$, and the next hop on the shortest paths from nodes $n_2$, $n_3$ and $n_4$ to $n_6$ is $n_5$. Arrows in the figure indicate the next hops on alternate routes. If nodes make unconstrained use of alternate paths, a routing loop may develop between nodes $n_2$, $n_3$ and $n_4$.

Since a source routing scheme uses explicit routes to transfer packets, it avoids routing loop problem. Breslau developed a comprehensive alternate routing architecture based on source routing for alternate paths. In the mode proposed, sources select alternate routes based on load information that the network distributes in a limited fashion. The results indicate that this architecture can improve throughput, setup delay and route quality. He also extended the benefits of trunk reservation in circuit-switched networks to the use of alternate paths in data networks. Breslau started his work at the earlier period of Internet. At that time the Internet was small, so source routing was then possible. However, this is not possible for current Internet.
Multipath routing in packet-switched networks have also been presented in many papers to solve network congestion problem. For example, Nelson et al. proposed a scheme that enables the use of multiple paths between source and destinations [67]. In their algorithm, multiple routes between each source and destination are computed based on hop count. These routes include both minimum hop paths as well as paths one hop longer than the minimum. Nodes make routing decisions for individual packets based on the current delay along different routes. Routing loops must be prevented when using paths that are longer than the minimum hop paths. This is accomplished by permitting only one node to make an alternate routing decision for each packet. Once a packet has been forwarded on an alternate route, it is tagged and all subsequent nodes must select the minimum hop path. Attar presented a distributed dynamic multipath scheme to enhance the single path routing [6]. In the scheme, nodes compute several routes to each destination using link state advertisements that are flooded to all network nodes. Routes are ranked as best, second best, third best, and so on. Data packets are tagged to denote the route they use, and intermediate nodes use this tag to make the proper forwarding decision. This level of coordination is required to prevent routing loops which would otherwise occur if routes longer than the minimum cost routes are used with hop-by-hop routing.

A more flexible algorithm for alternate path routing is presented by Harshavardhana [43]. The shortest paths are computed based on hop counts and nodes are classified by the number of hops they are away from a destination. A node can make an alternate routing decision to forward a packet to another node in the same class if certain conditions are met. These conditions involve the weight of links to that neighbor, and between the neighbor and its next hop on the shortest path. This allows more than one node along a path to make an alternate routing decision while avoiding routing loops. Nodes make routing decisions based on local congestion information;
alternate routes are only used if the next link on the shortest path is congested.

Wang presented an algorithm called Shortest Path First with Emergency Exists (SPF-EE) to solve dynamic routing and congestion control problems [91]. The SPF-EE algorithm allows local and temporary alternation of routes without global route updating. In the SPF-EE, the shortest paths are calculated based on the average link distance over a long time period and deal with momentary fluctuation with alternate paths. Alternate routing decisions are based on the length of local queues. The packets forwarded on alternate paths are tagged and only one alternate routing decision is allowed to avoid routing loops.

Murthy and Garcia-Luna-Aceves presented a framework for the modeling of multipath routing in connectionless networks that dynamically adapts to network congestion [66]. They adopted a leaky bucket scheduling mechanism to provide delay guarantees in the packet-switch network. Multiple loop-free paths from each node to a destination are maintained by means of a shortest multipath routing algorithm, which is based on a distributed update algorithm presented in the paper [38]. Their work was the inspiration for this dissertation. They used destination-based routing scheme, and we made use of the leaky bucket scheduling mechanism to regulate traffic. The dissertation focuses on the bottleneck link on each of equal paths, and uses leaky bucket as a monitoring mechanism to guide each bottleneck bandwidth. Since the ability of the destination controlling the source in a timely manner decreases as the network rate increases, we adopt a prediction mechanism instead of sending tokens from destination to the source node.

3.3 Routing in Circuit-Switched Networks

As mentioned in the previous sections, CBR is similar to the Dynamic/Adaptive Routing in telephone networks. Dynamic routing, based on network state, has a
long history, especially in circuit-switched networks. Dynamic routing methods are
categorized into three types in the circuit-switched networks, e.g. time-dependent
routing (TDR), event-dependent routing (EDR), and state-dependent routing (SDR)
[5].

In the TDR methods, the routing tables are altered at a fixed point in time during
the day or week. The TDR routing tables are determined by considering the time
variation of traffic load in the network.

In EDR methods, the routing tables are updated locally on the basis of whether
connections succeed or fail on a given path choice. When a call set-up request is
received by a node, it is routed first to the shortest path. If it has sufficient available
resources, then the resources are reserved on this link. Otherwise, the call set-up
can be cranked back to the previous node or a failure is declared. Crankback allows
the previous node to select an alternate link. The alternate link is selected from a
set of available alternate paths according to the given EDR routing table rules. For
instance, a $k$-shortest-path algorithm can be used to determine $k$ alternate links from
a node with distinct initial links [86]. Some mechanisms must be implemented during
path computation or call setup to prevent looping.

Performance studies of the alternate routing methods showed that alternate rout­
ing improves the throughput when traffic load is relatively light, but adversely affects
the performance when traffic load is heavy. Crankback could further degrade the
performance under these conditions [29]. The problem with alternate routing is that
both direct routed (shortest path) and alternate routed calls compete for the same re­
source. At higher loads, allocating these resources to alternate routed calls results in
the displacement of the shortest routed calls and hence the alternate routing of these
calls. Many approaches have been proposed to limit the flow of alternate routed calls
under high traffic loads. Trunk reservation is a scheme whereby on each link a certain
bandwidth is reserved for shortest routed calls. FAR (Fixed Alternate Routing) is an example of using trunk reservation scheme [63]. Alternate routed calls are allowed on a trunk as long as the remaining trunk bandwidth is greater than the reserved capacity. Thus, alternate routed calls cannot totally displace the shortest routed calls on a trunk.

In the SDR methods, the routing tables are altered automatically according to the state of the network. Fundamentally, there are two aspects to SDR:

- Measuring and gathering network state information, and

- Computing routes based on the available information

In general, SDR methods calculate a path cost for each connection request based on various factors such as the load-state or congestion state of the links in the network. RTNR (Real-time Network Routing) is an example of distributed connection-by-connection SDR methods [3, 4]. RTNR was used in the AT&T long distance network to support voice, data and wideband services. The switches used in RTNR first select the direct trunk group between the originating switch and the terminating switch. When no direct trunks are available, the originating switch checks the availability and load conditions of all of the two-link paths to the terminating switch on a per call basis. If any of these two-link paths are available, the call is set up over the least loaded two-link path. Traffic loads are dynamically balanced across trunks throughout the network to maximize the call throughput of the network. Link utilization is mapped into six discrete classes based on idle link virtual trunks. RTNR also used trunk reservation to reduce the chances that the link was used for two-link connections for calls to or from other switches; this enables the link to carry more direct traffic and therefore better handle the call load between the switches connected by the link.
Routing in circuit-switched networks is similar to the problem of routing real-time traffic in packet-switched networks. In both cases, routing must find a sequence of links with sufficient resources to carry performance sensitive traffic. In the case of circuit-switching, the resource is a dedicated circuit from source to destination, while routing in packet-switching, networks must find a path with sufficient bandwidth, processing capacity and buffer space to meet application performance requirement.

3.4 Routing in High Speed Networks

Multipath routing problem is also addressed in high-speed networks. Bahk and Zarki proposed a multipath source routing scheme to prevent the over-utilization of network resources by distributing the load at the beginning of congestion [7]. In their environment, dynamic information is distributed globally. Admission control is performed implicitly by source nodes rather than explicitly within the network. Using dynamic information, a source can decide whether a new session can use a network link. Hence, their route selection algorithm only needs to select one route to use, if any is available. This scheme depends on the fast and frequent global distribution of dynamic information. Their approach may be appropriate for a small homogeneous network, such as a single long haul backbone network. However, heterogeneity and scale make it inappropriate for large networks or Internet. In [79], resource reservation was made in parallel along several routes to control bursty traffic in high-speed networks, and resulted in increasing the probability to succeed in the reservation process and choosing the best one among several routes.

Max-min fair share, which is used in ATM ABR traffic management algorithms, fairly allocates the resources in networks based on virtual circuits (VC) or connections. Ma et. al. made use of congestion control information, i.e. max-min rate, as a routing metric to improve the throughput of high-bandwidth traffic in High Speed Networks.
3.5 Real-time Services

Providing real-time service in a packet switched network has received considerable attention in the literature. Most work in this area has concentrated on specifying packet scheduling algorithms, admission control algorithms, reservation protocols and flow specifications (see for example in [16, 19, 27, 35, 48, 49, 81, 82, 95]).

Integrated Services Packet Network (ISPN) was first proposed by Clark et.al. to describe a network providing different kind of services for real-time and datagram traffic [27]. They presented a ISPN architecture that supports two distinct kinds of real-time service: guaranteed service and predicted service. The guaranteed service supports applications requiring fixed delay bound and the predicted service supports applications requiring probabilistic delay bound.

To support real-time services in Internet (especially for IP environment) the Resource Reservation Protocol (RSVP) [17] has been advanced as the signaling protocol to enable network resources to be reserved for a connectionless data stream. RSVP is a receiver-driven protocol, i.e., the receiver of a data flow initials and maintains the resource reservation used for the flow. Each RSVP sender host transmits RSVP "Path" messages downstream along the uni-/multicast routes provided by the routing protocol(s), following the paths of the data. These Path messages store "path state" in each node along the way. This path state includes at least the unicast IP address of the previous hop node, which is used to route the Resv messages hop-by-hop in the reverse direction. Each receiver host sends RSVP reservation request (Resv) messages upstream towards the senders. These messages must follow exactly the reverse of the path(s) the data packets will use, upstream to all the sender hosts included in the sender selection. They create and maintain "reservation state" (link bandwidth and
buffer space) in each node along the path(s). Resv messages must finally be delivered to the sender hosts themselves, so that the hosts can set up appropriate traffic control parameters for the first hop.

While RSVP provide a method for requesting and reserving network resources, they do not provide a mechanism for determining a network path that has adequate resources to accommodate the requested QoS. Conversely, QoS-based routing allows the determination of a path that has a good chance of accommodating the requested QoS, but it does not include a mechanism to reserve the required resources.

Integrated services is implemented by four components: the signaling protocol (e.g. RSVP), the admission control routine, the classifier and the packet scheduler. Applications requiring guaranteed service or predictive service must set up the paths and reserve resources before transmitting their data. The admission control routines will decide whether a request for resources can be granted. When a router receives a packet, the classifier will perform a Multi-Field (MF) classification and put the packet in a specific queue based on the classification result. The packet scheduler will then schedule the packet accordingly to meet its QoS requirements.

Should admission control and resource reservation have to be adopted in Internet for supporting real-time applications? The question of whether admission control being implemented in Internet is discussed by Shenker [82]. His analysis suggested that for a network with only traditional data applications, efficacy is maximized by accepting all flows. However, when there are real-time applications, efficacy is maximized when some flows are turned away, which means that these flows are rejected by admission control. Furthermore, Breslau and Shenker present that in some circumstances reservation-capable networks have significant advantages over best-effort-only networks [81, 19].

The Integrate Services/RSVP architecture is influenced by the work of Farrar et
al. [34]. It represents a fundamental change to the current Internet architecture, which is founded on the concept that all flow-related state information should be in the end systems. Before the transmission of packets of a connection can begin, a channel based on the constraints must be established. A channel’s traffic is characterized by its peak rate, average rate, an averaging interval and a maximum packet size. The possible performance parameters include end-to-end packet delay, delay-jitter, buffer overflow probability and delay bound violation probability. The one of first wide area packet-switched networks to provide end-to-end per-connection performance guarantees is called Sequoia 2000 network [9]. Sequoia 2000 network employs the Tenet protocols to support the coexistence of computer data and real-time multimedia traffic. In the Tenet scheme, there are two levels of control: connection admission control at the connection level, and service discipline at the packet level. Before communication starts, the client specifies its traffic characteristics and performance requirements to the network. The client’s traffic and performance parameters are translated into local parameters, and a set of connection admission control conditions are tested at each switch. The new channel is accepted only if its admission would not cause the performance guarantees made to other channels to be violated. During date transfers, each switch will service packets from different channels according to a packet service discipline; by ensuring that the local performance requirements are met at each switch, the end-to-end performance requirements can be satisfied.

The problems with the Integrated Services architecture are: 1) the amount of state information increases proportionally with the number of flows. This places a huge storage and processing overhead on the routers. Therefore, this architecture does not scale well in the Internet core; 2) the requirement on routers is high. All routers must implement RSVP, admission control, MF classification and packet scheduling; 3) ubiquitous deployment is required for the guaranteed service. Incremental deployment
of the predictive service is possible by deploying RSVP functionality at the bottleneck nodes of a domain and tunneling the RSVP messages over other part of the domain.

Because of the difficulty in implementing and deploying Integrate Services and RSVP, Differentiated Services is introduced in IETF (Internet Engineering Task Force)[10, 15]. By using the Differentiated Services, network service providers can offer differing levels of network service to different traffic, in providing QoS to their customers. The basic premise of diff-serv networks is that routers within the networks handle packets different traffic flows by applying different per-hop behaviors (PHBs). The PHBs to be applied is specified by a code (the diff-serv code-point or DSCP) in the IP header of each packet. The advantage of such a scheme is that many traffic flows can be aggregated to one of a small number of PHBs, thereby simplifying the processing and storage associated with packet classification. In addition, there is no signaling state or related processing required in the diff-serv network since QoS is invoked on a packet-by-packet basis. QoS schemes such as Integrated Services/RSVP and Differentiated Services essentially provide graceful degradation of performance when traffic load is heavy. However, to avoid congestion at the first place, Traffic Engineer is motivated.

3.6 Traffic Engineering

Traffic engineering refers to the process of selecting the paths chosen by data traffic in order to balance the traffic load on the various links, routers, and switches in the network. Traffic engineering is most important in networks where multiple parallel or alternate paths are available. Traffic engineering is difficult to accomplish with datagram routing. Some degree of load balancing can be obtained by adjusting the metrics associated with network links. However, there is a limit as to how much can be accomplished in this way. In networks with a large number of alternative paths.
between any two points, balancing of the traffic levels on all links is difficult to achieve solely by adjustment of the metrics used with hop-by-hop datagram routing.

A widely utilized technique which divides traffic equally among the available equal paths is the ECMP (Equal Cost Multipath) method [45]. The ECMP is completely stable, since it does not make dynamic adjustments to the link cost based on loading. By using ECMP, a packet can be forwarded based on round robin, or according to a hash function applied to the source and destination pair. The round robin forwarding method is only applicable if the delays on the multiple paths are almost equal. Otherwise the application performance is degraded. For the non-real time traffic, delay differences greater than three times the packet serialization time can cause terrible TCP performance [88]. For the real-time traffic, delay differences on each equal path produce high delay jitter. On the other hand, since multipath based on the link cost cannot have an equal available bandwidth, this equal splitting is not optimal. To overcome the inefficiency of the EMCP method, Villamizar proposed a multipath method called OSPF Optimized Multipath (OSPF-OMP) [88]. In his method, traffic forwarding is adjusted based on link load. OMP provides a fine granularity of forwarding adjustment by flooding information within an OSPF area using Opaque LSAs.

Multiprotocol Label Switching (MPLS) is a standard in IETF [21]. MPLS is a forwarding scheme, motivated by using a fixed length label to decide packet handling. MPLS is a useful tool for Traffic Engineering. In the OSI seven-layer model, MPLS is between layer 2 and layer 3. Each MPLS packet has a header, which contains a 20-bit label, a 3-bit Class of Service (COS) field, an 1-bit label stack indicator and an 8-bit TTL field. Packets are classified and routed at the ingress Label Switched Paths (LSPs) of a MPLS-capable domain. MPLS headers are then inserted. When a LSR receives a labels packet, it will use the label as the index to look up the forwarding
table. This is faster than the process of parsing the routing table in search of the longest match done in IP routing. This label-switching process is similar to ATM’s VCI/VPI processing. Inside a MPLS domain, packet forwarding, classification and QoS service are determined by the labels and the COS fields. This makes core LSRs simple. Before a packet leaves a MPLS domain, its MPLS label is removed.

3.7 Conclusions

It is clear from the foregoing related work that though a number of algorithms for delay-constrained least-cost routing have been developed, they have generally tended to concentrate purely on the optimization aspects of routing. For an algorithm to actually perform well in practice, it is necessary to also take into account factors such as overall network performance, possibility of out-of-date information in the routing tables and frequent changes in link parameters. QoS-based routing must extend the current routing paradigm in following basic ways:

1. It must be able to maximize the overall performance of the network without sacrificing the requirements of any particular applications.

2. It must enable a resource reservation to be built into the routing strategy.

3. It must consider multiple constraints which is required in the case of QoS routing.

4. Some of new classes of service will require the distribution of additional routing metrics, e.g. delay, and available bandwidth. One approach to distribute bandwidth information is to extend the link state advertisements of protocols such as OSPF. If any of these metrics change frequently, routing updates can become more frequent, thereby consuming network bandwidth and router CPU
cycles. A tradeoff must be made between the need for accurate information and the need to avoid frequent flooding of the link state advertisement.

To reduce the frequency of the link state advertisements, one possible way is to distribute them only when there are topology changes, or significant bandwidth changes, e.g., more than 50 percent or more than 10 Mbps [93]. A hold-down timer should always be used to limit the frequency of such advertisements.

Since transmission of state information across wide area networks takes a fair amount of time, routing algorithms must also be designed to be adaptive to changes in network characteristics and must be capable of working with out-of-date information.

5. Today's opportunistic routing will shift traffic from one path to another as soon as a "better" path is found. The traffic will be shifted even if the existing path can meet the service requirements of the existing traffic. If routing calculation is tried to frequently changing consumable resources (e.g. available bandwidth), this change will happen more often and can introduce routing oscillations as traffic shifts back and forth between alternate paths. Furthermore, frequently changing routes can increase the variation in the delay and jitter experienced by the end users. To reduce the oscillation, one way is to keep the original flows on the same path and route the new coming flows to the new path; this method is called route "pinning".

6. Today's optimal path routing algorithms do not support alternate routing. If the best existing path cannot admit a new flow, the associated traffic cannot be forwarded even if an adequate path exists. Therefore, multipath routing needs to be employed.

7. Routing in the Internet is currently based only on the destination address of a
packet. Many multicast routing protocols require routing based on the source and destination of a packet. The Integrated Services architecture and RSVP allow QoS determination for an individual flow between a source and a destination. This set of routing granularities presents a problem for QoS-based routing solutions.

If routing based only on destination address is considered, then an intermediate router will route all flows between different sources and a given destination along the same path. This is acceptable if the path has adequate capacity but a problem arises if there are multiple flows to a destination that exceed the capacity of the link. Therefore, new granularities need to be employed for the QoS-based routing.

8. The main function of the Internet is for transferring data files, and even though audio or video applications are a high demand in the Internet, data files such as email and web applications are still a main part of traffic running on the Internet. Internet routing is working well for data transmission, there is no reason to change the whole routing architecture, also it is very hard to implement a totally new routing architecture in the world wide scale. Hence, developing an extension of OSPF is preferable.
CHAPTER 4

THE CONSIDERATIONS OF MULTIPLE METRICS AND MULTIPLE PATHS

The QoS-based routing requires the link metric to be extended to capture more network characteristics. On the other hand, almost all routing protocol, such as OSPF, IS-IS, etc., can form multiple equal cost paths between nodes. If traffic flows can utilize the multiple equal paths, the problem of overloading on the bottleneck link can be avoided.

In this chapter the possible metrics and feasible multipath routing scheme, which can be used in constructing new multipath routing architecture, are analyzed.

4.1 Multiple Metrics

As described in the previous chapters, many parameters can be used to characterize network resources, such as bandwidth, loss probability, delay, delay jitter, cost, etc. However, it may not be feasible to have these parameters as metrics. The most possible metrics for routing are delay and bandwidth. Through following definitions of delay, delay jitter, loss probability, etc, we explain why the bandwidth and delay are two important parameters.

1. Bounded end-to-end delay from source to destination is the one of the important QoS requirements, which is a cumulative result of the delay in each link that the packet travels. The delay on each path mainly consists of four components:

   - processing delay, that is between the time the packet is correctly received at the head node of the link and the time the packet is assigned to an
outgoing link queue for transmission.

- queueing delay, that is between the time the packet is assigned to a queue for transmission and the time it starts being transmitted. During this time, the packet waits while other packets in the transmission queue are transmitted.

- transmission delay, that is between the times that the first and last bits of the packet are transmitted.

- propagation delay, that is between the times that the last bit is transmitted at the head node of the link and the time the last bit is received at the tail node. This is proportional to the physical distance between transmitter and receiver; it can be relatively substantial, particularly for a satellite link or a very high speed link.

2. The processing delay is decided by computing speed in the router; it is independent of network load. The propagation delay is constant for each link and also independent of network traffic load. The transmission delay is calculated as a packet size divided by the bandwidth; it can be decided by looking up a table indexed by packet length and link speed. The queueing delay depends on the utilization of the link. Among the above four delays, the processing delay is very small, so it is relatively less important than other three delays. The queueing delay and transmission delay are determined by link utilization, or we can say by residual bandwidth; the propagation delay can be considered as a parallel parameter to the bandwidth. Therefore, the bandwidth and link propagation delay are two primary parameters.

3. The delay jitter is produced by the queueing delay that the packet experiences on each link along the path from source to destination; furthermore, the queueing
delay is determined by the bottleneck bandwidth (the minimal bandwidth along a path the packet travels from source to destination) and traffic characteristics. Thus, the delay jitter is reflected in the bandwidth metric.

4. The loss probability represents the packet drop rate. The reason for the packet drops is that the size of the packets waiting to transmit on a router larger than a buffer size in that router. Generally, the buffer size > \(2 \times \text{bandwidth} \times \text{delay}\) is expected to increase data transfer performance, so the only limiting factor becomes the true bandwidth of the network and not inadequate buffering.

5. The link cost is a general measurement. It can reflect the delay or the bandwidth; its value also can be chosen by a network administrator based on some rules in a local network area.

From the above analysis, we can see that the delay (accurately saying the propagation delay), bandwidth (residual bandwidth) should be considered as main parameters for QoS-based routing. There are two choices to construct path calculations based on the precedence of using the bandwidth and the propagation delay:

The first precedence is to use the bandwidth as the first level metric to calculate paths which satisfy an available bandwidth requirement, and then, if multiple paths exist, the path within the required propagation delay is chosen according to specific traffic flow.

The second precedence does the opposite of the first precedence; it first calculates equal paths whose propagation delay is within the required end-to-end delay bound and then among the equal paths, the path is chosen according to the required bottleneck bandwidth by the specific flow.

Since the bandwidth is easily changed, information about the available bandwidth may be disseminated very often to every node in a network; the routing overhead may be increased. Therefore, it is not proper to use bandwidth as the first level metric.
The propagation delay, which is related to the physical distance of each link, is a stable parameter. Thus, the second method is chosen for the routing algorithm in the dissertation.

4.2 Multiple Paths

When designing multipath routing, two important aspects need to be considered. One aspect is how to compute multiple paths and another is how to split traffic among the multiple paths. This section illustrates these two aspects.

4.2.1 Multiple Paths Availability Consideration

OSPF may form multiple equal paths between nodes according to the best path criteria. Multiple paths can also be obtained by using alternate paths which provides longer propagation delay than the best path. However, as analyzed in Chapter 3, any approach for the use of alternate paths in a hop-by-hop environment must severely constrain the use of alternate paths to avoid routing loops. Here, we only consider the availability of equal paths by using the other two ways: practical router configuration and relaxing the best path criteria.

Practical Router Configuration

Networks running OSPF are often heavily loaded. To satisfy the demands of bandwidth, topologies often evolve to include multiple paths. On the other hand, multiple paths may be initially designed to provide redundancy, but also result from incremental addition of circuits to accommodate traffic growth.

In general, there are two typical router configurations in the core networks on the Internet. The two router configurations, called config.1 config.2, are shown in Figure 4.1. Config.1 has 4 routers A, B, C, and D. If all links have the same distance, then the router A would have two equal paths to router D. Of course, we assume that the
four routers are in the same OSPF intra-domain.

Config.2 has three routers A, C and D, and would be a 1 hop equal path from router A to router C or router D. This type of configuration might be used if A is the main router that resides in a stub area (the area only receives input traffic), and C and D are Area Border Router(s) to the backbone to ensure redundancy to the backbone.

![Diagram](network_e.png)

Figure 4.1: Practical Router Configuration

Relaxing the Best Path Criteria

OSPF requires that only the best path be considered. For the purpose of increasing multiple equal paths, this criteria can be relaxed to allow a greater number of paths but not to the point of creating routing loops. In Figure 4.2, the number on each link is the link propagation delay. We calculate the best path based on the link propagation delay. The best path from node $n_1$ to node $n_6$ is $n_1 - n_2 - n_5 - n_6$ with a total propagation delay of 10.5 msec. We relax the best path criteria at node $n_2$, so that all paths whose propagation delay difference are less than 1 msec can be treated as equal paths. Therefore, we obtain two equal paths $n_1 - n_2 - n_5 - n_6$ and $n_1 - n_2 - n_4 - n_5 - n_6$ from $n_1$ to $n_6$. The two paths have propagation delay 10.5
msec and 11.0 msec, respectively. Furthermore, if we relax the criteria at $n_2$ so that the difference of propagation delay of equal paths is less than or equal to 1.5 msec, we obtain three equal paths. In addition to the above two equal paths, the third path is $n_1 - n_2 - n_3 - n_5 - n_6$ with propagation delay of 12.0 msec.

Figure 4.2: An Example of Relaxing the Best Path Criteria

4.2.2 Traffic Forwarding Consideration

When using multipath routing, the main issue which needs to be addressed is about how to split traffic load among the equal paths. In the source/destination forwarding method, traffic between any given source and destination remains on the same path. Routing the traffic from the same source to the same path is acceptable if the path has adequate capacity; however, a problem arises when there are multiple flows to a destination that exceed the capacity of the link. In the following, we analyze a specific situation in source/destination routing by simulation. When many real-time flows originate from the same source, they are routed to the same bottleneck link and experience high queueing delay and packet drops. The situation that the traffic also includes TCP flows is not considered here. The reason is that the TCP flow and UDP flow are totally different; it is assumed that the scheduling mechanism

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
has been used to distinguish both flows in the source router.

Let us take a look at a simple example. The simulation models of a 10-node network is shown in Figure 4.3.

![Figure 4.3: A Network Topology with Ten Nodes](image)

Each link has 640kbps bandwidth and 10msec propagation delay. The buffer size for each node is set to 50 packets. The ON/OFF traffic model is used to simulate real-time traffic. ON and OFF times are exponentially distributed. The packet burst time is 100ms, idle time is 10ms and the peak rate is 100kbps. The routing is dynamic with an update period of 4.0 seconds. The simulation runs 20 seconds. It should be noted that the topology and value for each parameter (bandwidth, delay, buffer size, etc.) are selected arbitrarily. Choosing small bandwidth for each link and high peak rate for traffic flow can reduce the amount of traffic flows, so that the result is easier to understand and explain compared to that of using large amount of traffic flows.

In the initial running period, four sessions of traffic start at \( n_1 \) and end at \( n_3 \), and two sessions of traffic start at \( n_2 \) and end at \( n_4 \), respectively. These six sessions of traffic begin at 0.1 second and end at 15.0 seconds. After the fifth second, traffic flows from nodes \( s_1, s_2, s_3 \) and \( s_4 \) start to transmit. At this time, the loading of link \((n_1, n_3)\) and link \((n_2, n_4)\) are changed to be 366.24kbps and 189.84kbps. Other links

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
are zero loading. Therefore, link \((n_1, n_3)\) and \((n_2, n_4)\) are bottleneck links. The link \((n_1, n_3)\) is more heavily loaded than link \((n_2, n_4)\). We set four cases of traffic patterns originated at nodes \(s_1, s_2, s_3,\) and \(s_4\).

- Case 1, four sessions of traffic originate from node \(s_1\), and are routed to link \((n_1, n_3)\).
- Case 2, four sessions of traffic originate from node \(s_1\), and are routed to link \((n_2, n_4)\).
- Case 3, four sessions of traffic originate from nodes \(s_1, s_2, s_3\) and \(s_4\). Two sessions are routed to link \((n_1, n_3)\), and the other two sessions of traffic are routed to link \((n_2, n_4)\).
- Case 4, four sessions of traffic originate from nodes \(s_1, s_2, s_3\) and \(s_4\). One session is routed to link \((n_1, n_3)\), and the other three sessions of traffic are routed to link \((n_2, n_4)\).

The result is shown in the Table 4.1. In using the source/destination to route traffic, when large flows come from the same source, they still experience high delay and packet drops as in Case 1. In Case 2, flows go to the lightly loaded path, so there is much less delay. Case 3 splits traffic equally; the result is also much better than Case 1. Case 4 simply shows the condition when traffic can be forwarded based on link loading, which is similar to Villamizer's method [88]. The question is how to
make flows avoid heavily loaded links when flows come from the same source node. It is impossible to give any source node a high (or low) priority and forward its traffic to the lightly (or heavily) loaded link.

Flow-based routing has the benefit for adjusting traffic according to the requirement of each individual flow. The previous chapter introduces MPLS, which provides a labeling mechanism to make flow-based hop-by-hop routing possible [21]. In MPLS, a label is put on each traffic flow or aggregation flow, so that the router can route the packets belonging to the same traffic to the same route. However, since the number of traffic flows getting into a node in the Internet is very high, flow-based routing has scalable problems. Differentiated Services mechanism can offer differing levels of network service to different traffic. As introduced in Chapter 2, Differentiated Service provides two basic services in addition to the Best-Effort Services; the two services are Premium Service and Assured Service.

Premium Service is used for applications requiring low delay and low jitter service, such as video broadcasts, voice-over-IP, etc. Premium Service needs performance guaranteed, therefore, resource reservation is needed for its transmission. Premium rates might be configured on a subscription basis in the near-term, or on-demand when dynamic set-up or signaling is available. On the other hand, Assured Service is used for applications requiring higher reliability than Best Effort Service. This service may be provided by ISP to some individual customers who want an assurance that IP packets are forwarded with high probability, for example, when an company uses the Internet to interconnect its geographically distributed sites.

Since service is allocated in the granularity of a class, the amount of state information is proportional to the number of classes rather than the number of flows. Therefore, Differentiated Services is more scalable; as a result, routing based on Differentiated Services is also scalable. Based on these classes, a multipath routing
scheme will be derived in the following chapter.
The most important goal for routing in the Internet is that the routing architecture enable high throughput and reduce experienced end-to-end delay or delay jitter for real-time traffic. However, routing by itself cannot guarantee high throughput and end-to-end delay in a multimedia environment without incorporating scheduling and admission control algorithms, which are independent of the routing algorithm. We note, however, given the existence of particular scheduling and admission control algorithms, routing can affect throughput by its choice of links used to transfer the traffic [18]. Furthermore, an efficient routing scheme can reduce queuing delay that packets experience when waiting in the queue. This chapter exclusively focuses on describing a multipath routing scheme, which is called Two-level Class-based Routing with Prediction (TCMP). The TCMP is designed to meet the goal of QoS-based routing. The TCMP intends to support two types of services: Premium Service and Assured Service [10, 44, 68]. The guaranteed service in Premium Service can be supported by resource reservation. The requirements of different levels of drop precedence in Assured Service can be satisfied by choosing a corresponding path among the equal paths.

The use of dynamic information can improve network performance by balancing the load across network links and reducing the delay encountered in route setup. However, if the routing information is changed so often, the network is prone to oscillation. On the other hand, frequently updating routing information consumes link bandwidth and router's processing time. Furthermore, the transmission of link
state information across wide area networks takes a fair amount of time. The TCMP is designed to be adaptive to changes in network characteristics and be capable of working with out-of-date information.

The basic principle behind TCMP scheme is to control traffic forwarding by using the traffic load information monitored in previous stable period. The path selection function can utilize resources on multiple paths to increase network throughput and reduce the queueing delay. Overall, the key features of the proposed routing architecture are:

1. It uses distributed, or hop-by-hop, multipath routing algorithms.

2. On the first-level routing table, multiple routes are computed based on propagation delay between each source and destination, so that the end-to-end delay requirements can be approximately satisfied.

3. On the second-level routing table, the bottleneck bandwidth on each of the equal delay routes is obtained, so that the delay-jitter or queueing delay can be reduced.

4. Information about the link load is periodically distributed to network nodes.

5. Leaky buckets are used as guidance for the bottleneck bandwidth of each equal path to control packet forwarding at each node, and further support resource reservation.

The propagation delay, which is used in calculating the first-level routing table, is between the times that the last bit is transmitted at the head node of the link and the time the last bit is received at the tail node. This is proportional to the physical distance between transmitter and receiver; it can be relatively substantial, particularly for a satellite link or a very high speed link. The TCMP uses the propagation delay as
a metric to exclude some high delay link, so that the delay difference for the splitting traffic is not very big. In later sections, it will be clear that the TCMP can generally keep packets for a traffic to travel through the same route; however, it is still possible for a traffic to be split when there is a major change in the bottleneck bandwidth.

5.1 Routing Scheme Description

In a router, there are two main parts, one is background code and the other is a forwarding path part. The simple model in a router is shown in the Figure 5.1. The background code, which includes Routing Table Construction and Traffic Control routine, creates data structures that control the Forwarding Path. The Routing Table Construction routine implements a particular routing protocol and builds a routing database. Traffic Control contains three agents, i.e. Reservation Setup, Admission Control and Management. The Reservation Setup agent implements the protocol used to set up resource reservations. If Admission Control gives the permission for a new session, the appropriate changes are made to the classifier and packet scheduler database to implement the desired QoS. Finally, every router supports an agent for network management. The agent must be able to modify the classifier and packet scheduler database to set up controlled link-sharing and to set admission control. The forwarding path of the router is executed for every packet. Internet forwarding interprets the internetworking protocol header appropriate to the protocol suite. For each packet, a forwarder executes a suite-dependent classifier and then passes the packet and its class to the appropriate output driver. The output driver implements the packet scheduler.

The TCMP does not regulate how a traffic flow is transmitted when it is routed to a path; this task is left to the packet scheduling mechanism. Here it is assumed that the FIFO (First In First Out) scheme is adopted. However, the TCMP supports
resource reservation setup by providing the load information on the bottleneck link along each of the multipath.

In the TCMP routing scheme, routing is done on a per destination basis over multiple paths. The routing table does not only contain the next hop for each specific destination, but also contains the information about network load for guiding packet forwarding and resource reservation. To forward the packets to a given destination, the TCMP uses two routing metrics: a first-level metric based on the link propagation delay from a source to all of its destinations in the network, and a second-level metric based on the bottleneck bandwidth along the path.

In the TCMP, routing is done on a hop-by-hop basis independently at each node. Each node monitors traffic on the incoming and outgoing links periodically. Given the capacity of each link, the utilization of the link can be determined. Based on the utilization of the outgoing link for a node and the contribution of traffic load coming from each neighbor, a credit is computed for the outgoing link and given to each of the upstream links. The bottleneck bandwidth link along each path is then determined. Each time the network state changes, paths are recomputed and the new network state is obtained. This is made possible by the periodic exchange of routing
information.

At any given time, each node maintains a database which describes the topology of the network, the delay and bandwidth of each link, as measured by itself or reported by the nodes to which it connects. Each node maintains a routing table, a neighbor's credit table, and a packet forward guiding table. The node's routing table contains the routing information about the shortest multipath to all destinations and the available bandwidth corresponding to each of the multipath. The neighbor's credit table for a node contains credit for all links seen from its neighbors to the node. The available bandwidth of the link for a node then is determined by the credit and the link capacity. The packet forward guiding table provides the bucket size for each next hop. There is a threshold for the bucket size, and the token rate for the bucket is generated based on the available bandwidth on the bottleneck link.

5.2 Neighbor Monitoring Table

In the connectionless network, all the nodes along any path from a source to a given destination can contribute to the flow to that destination. We have to consider the effect of the flows coming from other nodes, rather than only the nodes along the equal multiple paths.

For a general case, Figure 5.2 shows a node \( n_k \) in a network; arrows mean that flows go to \( n_j \) from nodes \( n_i, \; i = 1, 2, ..., k \). The neighbors set is \( NB \). \( |NB| \) denotes the number of elements in \( NB \). \( NB = n_1, n_2, ..., n_{|NB|} \). \( p^i_{kj} \) denotes the traffic load from incoming node \( n_i \) to outgoing node \( n_j \) through node \( n_k \). \( C^i_{kj} \) denotes the credit that outgoing link \((k, j)\) gives to the incoming link \((i, k)\). A node calculates a neighbor's credit based on its monitoring of the incoming traffic load from the neighbor to a specific outgoing link. The algorithm is described in Figure 5.3. To simplify notation, there is no distinction between 1 and \( n_1 \), \( i \) and \( n_i \), and so on.
Credit information at each node is updated periodically. On initialization, credit at each node is equally distributed among neighbors and itself. The credit is dynamically assigned thereafter among the node itself and all the active neighbor links, depending on the traffic which come from the upstream neighbors or originated by the node itself. The backup credit \( \Delta C \) is given for each node in case of more traffic flow coming to the node from its neighbors than that in the last stable interval. The backup credit \( \Delta C \) is also given to each upstream link, so that more traffic is transmitted from the upstream link than that in the last stable period. When the traffic load is less than the predicted one, it will deduct \( \Delta C \) credit from the original value. On the other hand, if traffic load is more than the predicted one, it will add \( \Delta C \) to the original credit. If there is no change, the credit will be kept the same. Finally when the total credit at a node is more than one, e.g. the link is congested, the credit for each neighbor and itself is set to initial value. The credit table is given in Table 5.1. In the table, \( C_{kj}^0 \) denotes the credit given by link \((k, j)\) for the node \( n_k \) itself.

Taking Figure 5.2 as an example, if the node \( k \) has four neighbors, e.g. \( j = 4 \), on initialization, the credit given from its neighbor \( n_4 \), e.g. link \((k, j)\) to other neighbors and itself can be calculated as:

![Figure 5.2: Neighboring Structure of Node \( n_k \)](image-url)
Variable:
\( l_{ij} \): the capacity of link \((k, j)\)
\( \rho_{ij} \): the monitored traffic load coming from \(i\) to \(j\) through \(k\)
\( C_{kj}^i \): the permitted credit for \(i\) given by link \((k, j)\)
\( \Delta C \): backup credit

Procedure Initialize:
when router \(k\) initializes itself
begin
\[ C_{kj}^i := \frac{100}{|N|} - \Delta C \]
end

Running Period:
At each update time
begin
\( \tilde{C}_{kj}^i := \frac{\rho_{ij}}{l_{ij}} \)
if \( \tilde{C}_{kj}^i < C_{kj}^i \leq C_{kj}^i + \Delta C \)
begin
\[ C_{kj}^i := \tilde{C}_{kj}^i + \Delta C \]
end
elseif \( \tilde{C}_{kj}^i < C_{kj}^i \)
begin
\[ C_{kj}^i := \tilde{C}_{kj}^i - \Delta C \]
end
else
\[ C_{kj}^i := \tilde{C}_{kj}^i \]
if \( \sum_{j \in N_k} (C_{kj}^i + \Delta C) > 1 \)
begin
\[ C_{kj}^i := \frac{100}{|N|} - \Delta C \]
end
end

Figure 5.3: The Algorithm for Computing Neighbor's Credits
Table 5.1: Neighbor’s Credit Calculation Table

\[
C_{k4}^1 = C_{k4}^2 = C_{k4}^3 = C_{k4}^0 = \frac{\text{capacity of } \text{link}(k,j)}{4}
\]

5.3 Routing Table Construction

The algorithm for building a routing table is described in Figure 5.4. The algorithm for computing the first-level multipath is similar to Dijkstra algorithm; however, the narrowest link (bottleneck link) along a path is recorded. The second-level routing just sorts the multiple paths according to the bottleneck bandwidth. \( N \) is the set of nodes in a network, and \( M \) is the set of nodes for which the shortest paths have not been found. For any one of source nodes \( s \), \( d^*_j \) is defined as the propagation delay from \( s \) to any destination node \( j \). The algorithm maintains the information about the equal propagation delay paths in the routing table. \( E^*_j \) is the set of equal paths from \( s \) to \( j \); \( abw^*_j \) is the narrowest available bandwidth along each of the multipath from \( s \) to \( j \); \( mp^*_j \) maintains the number of equal paths from \( s \) to \( j \). Initially, \( M = N \) and all \( d^*_j = \infty \) for \( j \neq s \). At each step of the algorithm, the node \( n_i \) in \( M \) with the smallest propagation delay \( d^*_i \) is removed from \( M \). Each neighbor of \( n_i \) in \( M \) is examined to see whether a path through \( n_i \) would be shorter than the currently shortest path. \( \text{prog}(n_k, n_i) \) and \( \text{BW}(n_k, n_i) \) are the propagation delay and bandwidth of link \((k, i)\), respectively. \( \text{BW}(n_k, n_i) \) is the measured value in the last stable period before the updating. \( abw(n_k, n_i) \) is the available bandwidth given to the incoming node \( p \) by link \((n_k, n_i)\) in the stable period. The amount of \( abw(n_k, n_i) \) is determined by credit
\[ \text{Table 5.2: Routing Table} \]

\begin{tabular}{|c|c|c|c|}
\hline
\text{dst} & \text{mpth} & \text{nxt-hop} & \text{abw} \\
\hline
\text{d}_1 & \text{mp}_1^k & \text{n}_{11} & \text{abw}_1^k(1) \\
 & & \text{n}_{12} & \text{abw}_1^k(2) \\
 & & \text{...} & \text{...} \\
 & & \text{n}_{1i} & \text{abw}_1^k(i) \\
\hline
\text{...} & \text{...} & \text{...} & \text{...} \\
\hline
\text{d}_j & \text{mp}_j^k & \text{n}_{j1} & \text{abw}_1^k(1) \\
 & & \text{n}_{j2} & \text{abw}_1^k(2) \\
 & & \text{...} & \text{...} \\
 & & \text{n}_{ji} & \text{abw}_1^k(i) \\
\hline
\end{tabular}

\( C_{ki}^p \) obtained in the neighbor's credit calculation table.

Finally, we have the routing table as given in Table 5.2 for any node \( n_k \) to all active destinations \( d_j \) in the network.

### 5.4 Packet Forwarding

The traffic at each node is forwarded based on the virtual leaky buckets, which is created based on the bottleneck bandwidth. The leaky bucket is called virtual, because it is not used for regulating the traffic as a scheduler. Traditionally, the leaky bucket scheme, which regulates the burstiness of the transmitted traffic, is used for traffic rate control as shown in Figure 5.5. To join the transmission queue, a packet must get a permit from the permit queue. A new permit is generated every \( 1/r \) seconds, where \( r \) is the desired input rate, as long as the number of permits does not exceed a given threshold \( W \). The buckets are session oriented.

In the TCMP routing method, the leaky buckets are destination oriented. The buckets are used for monitoring the bottleneck bandwidth along each of multipath. The leaky-bucket parameters are maintained for each bottleneck link based on each active destination rather than for each session. For a given destination \( j \), permitted tokens are created at a rate \( r_j^i \) at node \( i \), which is called the token generation rate.
Variables

$N$: the set of nodes

$M$: the set of nodes for which the shortest paths have not been found.

$d^*_j$: propagation delay from $s$ to $j$

$mp^*_j$: number of equal paths from $s$ to $j$

$E^*_j$: the set of equal paths from $s$ to $j$

$abw^*_j$: the narrowest bandwidth of one of equal paths from $i$ to $j$

Initialization

$M := N$

$d^*_j := \infty$ for $j \neq s$

$d^*_s := 0$

$mp^*_s := 1$

$abw^*_j(mp^*_j) := \infty$

$E^*_j := \phi$

Procedure

begin

for $j = 0$ to $|N| - 1$

find $n_k \in M$ with minimum $d^*_k$

if $n_k = n_s$ then $p := 0$

for each link $(n_k, n_i)$ with $n_i \in M$

if $d^*_k > d^*_k + prog(n_k, n_i)$

begin

$d^*_k := d^*_k + prog(n_k, n_i)$

$abw(n_k, n_i) := C^k_{\#i} \times BW(n_k, n_i)$

if $abw^*_j(mp^*_j) > abw(n_k, n_i)$

then $abw^*_j(mp^*_j) := abw(n_k, n_i)$

end

if $d^*_k = d^*_k + prog(n_k, n_i)$

begin

$E^*_j := E^*_j \cup \{n_k\}$

$mp^*_j := mp^*_j + 1$

if $abw^*_j(mp^*_j) > abw(n_k, n_i)$

then $abw^*_j(mp^*_j) := abw(n_k, n_i)$

end

endfor

$M := M - \{n_k\}$

$p := k$

endfor

end

Figure 5.4: Path Calculation Algorithm
\( r_j \) is decided by the bottleneck bandwidth \( abw_j \). The bucket size, denoted by \( w_j(t) \) gives the maximum number of packets that can be transmitted from \( i \) to \( j \) at time \( t \). Similar to [66], for the time \( t > 0 \), \( w_j(t) \) is defined as:

\[
w_j(t) = l_j(t) + Q_j(t)
\]

where \( l_j(t) \) is the number of left-over tokens in the bucket at node \( i \) for destination \( j \) at time \( t \), and \( Q_j(t) \) is the backlog for destination \( j \) at time \( t \). The backlog presents the packets waiting in the queue. \( w_j(t) \) has to be less than its threshold \( W \), which is determined by the buffer size available at the node \( s \). The number of packets sent along one of multiple paths depends on the permitted tokens in the leaky bucket. Figure 5.6 simply describes the basic idea of packet forwarding scheme in the TCMP.

It is assumed the traffic is classified as several classes based on their service requirements. Class 1 requires the lowest drop precedence for Assured Service, and Class 2 is on the second rank of the requirement, and so on. The bucket size for each of the equal paths is monitored at each sampling period \( T_{sample} \). \( T_{sample} \) is much less
than the updating period in the network. For any node \(i\) to its destination \(j\), at each time of sampling, the multipath is reordered based on the tokens left in its bucket; \(\text{order}_j^i[k]\) is equal to 1 for the path with largest bucket size. \(\tau_{\text{sample}}\) is reset at each sampling time. The function of monitoring is to keep higher classes of traffic to be always transferred to the path with higher bucket size. Since there is always traffic with different classes routed to different paths, network oscillation is avoided. The forwarding algorithm is shown in Figure 5.7. If the class of a flow is larger than the number of multipath, or the class is unknown, the flow is routed to the path with the lowest \(\text{order}_j^i[k]\).

One aspect to be considered is when a link \((i,k)\) is a bottleneck link from node \(s\) to several destinations, for example, \(j_1,j_2,j_3\), as shown in Figure 5.8. In this situation, the bucket in node \(i\) is for these destinations. The tokens in the bucket are changed according to packets forwarded to any one of the destinations.
Variables

- \( l^*_j \): the left-over token in the bucket from node \( i \) to destination \( j \)
- \( \text{order}^*_j \): the buckets is ordered from the largest bucket to lowest bucket
- \( \text{class}(f) \): the classification of the traffic flow based on its requirement
- \( mp^*_j \): number of equal paths
- \( L \): packet size

Procedure

At the sampling time \( \tau_{\text{sample}} \)

begin

comparing \( l^*_j[k] \), \( 1 \leq k \leq mp^*_j \)
mark the \( k \)th path with \( \text{order}^*_j[k] \)
if \( l^*_j[k] \leq 0 \) then \( mp^*_j := mp^*_j - 1 \)
reset \( \tau_{\text{sample}} \)
end

During traffic transmission

begin

c := \text{class}(f)
if \( c > mp^*_j \)

begin
transmit flow \( f \) by the path with \( \text{order}^*_j[k] = mp^*_j \)
\( l^*_j := l^*_j - L \)
end
else

begin transmit flow \( f \) by the path with \( \text{order}^*_j[k] = c \)
\( l^*_j := l^*_j - L \)
end
end

Figure 5.7: Packet Forwarding Algorithm

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
5.5 Resource Reservation

The Premium Service requires performance guaranteed for the applications; thus, resource reservation needs to be adopted. When choosing a path among equal paths to reserve, leaky bucket can give an approximate indication of the resource on each path. The path with the largest bucket size is first used for reservation. After a path is reserved, the amount of reserved bandwidth is deducted from the bucket of that path; the rest of tokens are used to guide the packet forwarding for Assured Service. Figure 5.9 shows the resource reservation algorithm. The reservation starts from source node $s$ to destination $j$. The path with the highest bucket size, e.g. $order_j^*[k] = 1$, is selected for reservation. If the required bandwidth $rbw_j$ is less than the available bandwidth $abw_j[k]$ on the $kth$ link, reservation is continued to the next hop $nh$. The left token in the bucket of the $kth$ link is recalculated and the multipath are reordered based on their current bucket size. If $rbw_j$ is larger than $abw_j[k]$ at any node $i$ between $s$ to $j$, $block$ is set to 1 and the reservation fails.
Variables

$s$: source node

$l_i^j$: the left-over token in the bucket from node $i$ to destination $j$

$\text{order}_j^i[k]$: the index from largest to the lowest bucket size of $k$th link.

$mp_j$: number of equal paths from $i$ to $j$

$rbw_j^i$: the bandwidth needs to be reserved for the traffic from $s$ destined to $j$

$abw_k^i$: the available bandwidth on the link $(i,k)$

Initialization

$block := 0$

$i := s$

Procedure

begin

while ($i \neq j$)

for $n = 0$ to $mp_j^i - 1$

find next hop $nh$, $(i,nh)$ is the $k$th link, so that $\text{order}_j^i[k] = 1$

if $abw_k^i > rbw_j^i$

begin

$l_j^i[k] := l_j^i[k] - rbw_j^i$

comparing $l_j^i[k]$, $1 \leq k \leq mp_j^i$

mark the $k$th path with $\text{order}_j^i[k]$

if $l_j^i[k] < 0$ then $mp_j^i := mp_j^i - 1$

reset $\tau_{\text{sample}}$

$i := nh$

end

close $block := 1$; $i := j$

endwhile

end

Figure 5.9: Resource Reservation Algorithm
5.6 Information Distribution

For the credit table of each node, there are two ways to disseminate the information to other nodes in the network. One way is to flood the information to all other nodes. Another way is that after one source node has calculated its routing table, it sends a message to the bottleneck node on its multipath, and then the bottleneck node sends its credit table to the source node.

The first flooding method is used in current OSPF protocol. By using the flooding method, the original node sends its information in the form of a packet to its neighbors, and then the neighbors relay the information to their neighbors, and so on, until the packet reaches all nodes in the network. The credits are used in a float number, which is 4 bytes long. For a network with \( n \) nodes and average degree \( m \), the credit table for one node has \( 4 \times n \times m \) bytes. For a high average degree network, this may produce a high information transmission load. However, since this information can be distributed with link-state information, it does not need a more specific process.

For the current Internet, each OSPF router originates one or more Link State Advertisements (LSAs) to describe its local part of the routing domain. Taken together, the LSAs form the link-state database and enable the routing calculations.

Another method avoids the large amount of link state traffic, but it produces more processing time, and it also needs extra header definition for identifying this traffic. Furthermore, sending a request message to a bottleneck link node and waiting on its feedback produces extra propagation delay.
CHAPTER 6

ANALYSIS OF THE TCMP ALGORITHM

Networks operating in a packet switched mode is very flexible and complicated. In the packet-switched network, routing loop is a fundamental problem when multipath routing scheme is employed. Furthermore, routing as a sophisticated distributed mechanisms, its performance is hard to be accurately analyzed, thus, worst-case analysis has to be applied. This chapter first shows that loop free can be maintained in TCMP scheme. Following that, worst-case bounds on delay and backlog are derived when the TCMP scheme is adopted.

6.1 Loop Freedom in TCMP

In the TCMP scheme, the first-level paths are obtained by computing the shortest multipath. There should be no routing loops.

**Theorem 6.1:** Multiple paths obtained by using Two-Level Class-Based Routing with Prediction (TCMP) are loop-free.

**Proof:** By contradiction. Figure 6.1 shows three nodes in any network topology. It is supposed that nodes $b$ and $c$ are involved in a loop for destination $d$. We denote $d_{ij}$ as the propagation delay from node $i$ to node $j$. By the definition of the equal paths with the shortest propagation delay, we have

\[ d_{bd} \leq d_{bc} + d_{cd} \quad (6.1) \]
From Inequalities 6.1 and 6.2, we have

\[ d_{cd} \leq d_{cb} + d_{bd} \]  

(6.2)

The above equality is true as long as \( d_{cb} + d_{bc} = 0 \), which is not the case in the network. This completes the proof.

6.2 Algorithm Complexity

In this section, we analyze the complexity of TCMP. Our comparison is made in terms of the number of steps of computation and number of messages required for TCMP to construct a routing table. We refer to the number of steps required by an algorithm as its computation complexity, and to the number of messages it requires as its communication complexity. We also consider the storage space required by TCMP algorithm, which is called storage complexity.
We are given a directed graph \((N, A)\) with node number 1, 2, ..., \(N\). Each arc \((i, j) \in A\) has a cost or "length" associated with it. \(|A|\) is the number of arcs.

1. Computation Complexity

TCMP route computation uses link-state Dijkstra’s algorithm. The best estimate of the worst case running time that has been obtained is \(O(|A| + N\log N)\) \([13]\).

For TCMP algorithm, the operation for finding the bottleneck bandwidth is included in \(O(A)\). In the Dijkstra algorithm, the \(O(A)\) operation is for arc examination. The operation for sorting equal paths in TCMP is \(O(D\log D)\) using sequential sorting algorithm, where \(D\) is the maximum degree of a node. Since \(D\) can never be bigger than \(N\), \(O(D\log D)\) does need to be considered. Therefore, the TCMP totally has computation complexity: \(O(|A| + N\log N)\).

2. Communication Complexity

For Dijkstra’s algorithm, to broadcast all arc lengths from some node to all other nodes over an optimally chosen spanning tree takes \(O(d + |A|)\), where \(d\) is the diameter of the network \([14]\). The TCMP has similar communication complexity to Dijkstra algorithm. The only difference is that the broadcast message needs also contain bandwidth information.

3. Storage Complexity

Original Dijkstra’s algorithm has \(O(N^2)\) storage complexity. In TCMP, each node maintains a routing table, a neighbor’s credit table, and a packet forward guiding table. The neighbor’s credit table and packet forwarding table are extra tables compared to Dijkstra’s algorithm; they have \(O(D^2)\), and \(O(DN)\) complexity, respectively. Therefore, TCMP has \(O(N^2 + D^2 + DN)\) storage complexity. The main reason for the higher storage complexity compared to...
Dijkstra's algorithm is that TCMP needs bandwidth information on each of multiple paths. Tables 5.1 and 5.2 in Chapter 5 illustrate this point clearly.

### 6.3 Worst Case Steady-State Delays

This section derives an approximate worst-case delay on the bottleneck link and end-to-end delay bound for the TCMP algorithm. In the first-level route calculation, the end-to-end delay is first limited by the link propagation delay. However, queuing delay on each node along the path can affect the end-to-end delay and also delay jitter, especially when the link is heavily loaded.

A similar approach as in [66] is adopted to derive approximate worst-case delay for each bottleneck link in a connectionless architecture. We assume that the topology is stable without link failure. The reason for "approximate worst-case delay" is that the TCMP utilizes the leaky bucket only as a monitoring mechanism as described previously, it cannot provide performance guaranteed. The performance guaranteed can be provided by resource reservation mechanism.

In a connectionless network where routes are computed in a distributed way, the path taken by a packet can change dynamically depending on the congestion level in the network. Routing is done on a hop-by-hop basis, independently at each router. Therefore, the total traffic at a node will be the sum of the traffic on all its links connecting to upstream neighbors. As in [66], to obtain an expression for the worst-case bound, we make the following assumptions.

1. The nodes send traffic flows to a node through a bottleneck link as long as tokens are available for the nodes to the bottleneck link.

2. At every node, traffic traversing toward the bottleneck link is treated independently for each equal path.

3. Traffic, which traverse down the link \((i,k)\) and routed to the destination \(j\) in...
the interval \([\tau, t]\), denoted by \(A_j[i, k](\tau, t)\) is the sum of the traffic coming from all upstream neighbors of node \(i\) traversing to \(j\), denoted by \(\sum_{n \in NB_i} f^n_j[i, k]\), and the traffic originated at the node \(i\) itself, denoted by \(f^n[i, k](\tau, t)\), i.e.,

\[
A_j[i, k](\tau, t) = f^n_j[i, k](\tau, t) + \sum_{n \in NB_i} f^n_j[i, k]
\]

(6.4)

where \(NB_i\) denotes a node set including all the neighbors of node \(i\). The flows coming from neighbors of node \(i\) includes the flows which are constrained by link \((i, k)\), i.e. link \((i, k)\) is bottleneck link, and the flows which traverse the path that \((i, k)\) is not a bottleneck link.

\[
\sum_{n \in NB_i} f^n_j[i, k] = \sum_{n \in NB_i} f^n_j[i, k] + \sum_{n \in NB_i} \bar{f}^n_j[i, k]
\]

(6.5)

\(\sum_{n \in NB_i} \bar{f}^n_j[i, k]\) denotes the flows having link \((i, k)\) as bottleneck link, and \(\sum_{n \in NB_i} f^n_j[i, k]\) denotes the flows constrained by other links instead of link \((i, k)\).

If the bottleneck link along one of equal path is not link \((i, k)\), the bottleneck link should have less available bandwidth and allow less flows to be transmitted than the link \((i, k)\) does. A routing variable \(\phi^n_j[i, k]\) is defined for the bottleneck link \((i, k)\) from any upstream nodes of node \(i\) to destination \(j\) as the ratio of the flows not taking link \((i, k)\) as bottleneck link with respect to the flows taking \((i, k)\) as bottleneck link.

\[
\phi^n_j[i, k] = \frac{\sum_{n \in NB_i} \bar{f}^n_j[i, k]}{\sum_{n \in NB_i} f^n_j[i, k]}
\]

(6.6)

where, \(\phi^n_j[i, k] < 1\). Similarly, for \(f^n[i, k](\tau, t)\), there are also two kinds of flows, one is the flow constrained by link \((i, k)\), another is the flow constrained by other link. As in Equation 6.5 and 6.6, we have

\[
f^n_j[i, k] = \bar{f}^n_j[i, k] + \bar{f}^n_j[i, k]
\]

(6.7)
According to above equations, Equation 6.4 can be rewritten as:

\[
\phi_j^i[i, k] = \frac{f_j^i[i, k]}{f_j^o[i, k]} \tag{6.8}
\]

The delay on a link \((i, k)\) (per hop delay) for the destination \(j\), denoted by \(d_j[i, k]\), is the sum of the queuing delay, transmission delay, and propagation delay. The delay, which is denoted by \(\delta_j[i, k]\), is the sum of transmission delay and propagation delay, and depends on the congestion level of the link as well as the link capacity. The queuing delay is the time a packet has to wait at a node before it is processed. The waiting time of a packet depends on the number of packets already present in the queue at the time a packet arrives. This is referred to as the backlog at node \(i\) for destination \(j\) and is denoted by \(Q_j^i\). Therefore, as in [66], the delay on link \((i, k)\) for destination \(j\) at time \(t\) is:

\[
d_j[i, k](t) = \delta_j[i, k](t) + Q_j^i(t) \times \delta_j[i, k](t) = \delta_j[i, k](t)[1 + Q_j^i(t)] \tag{6.10}
\]

The backlog number of packets for a given destination \(j\) at a given time \(t\) can be defined as the difference in the incoming traffic \(A_j^i(t)\) and the outgoing traffic \(S_j^i(t)\) at a node, i.e.,

\[
Q_j^i(t) = A_j^i(t) - S_j^i(t) \tag{6.11}
\]

In the following, the approximate bound on end-to-end delay is analyzed. First, the packet size is assumed to be negligible. Following that, the situation when packet size is non-negligible is considered.
6.3.1 Negligible Packet Size

**Theorem 6.2:** When packet size is negligible, TCMP can achieve an approximate end-to-end delay bound. The delay bound is determined by the maximum propagation delay along one of equal paths to a destination, the number of hops from source to destination, and bucket parameters.

**Proof:** When packet size is very small, the maximum packet transmission time at any link of the network is negligible. From equations obtained in the last session, for the time interval \((\tau, t)\), the maximum backlog number of packets for the link \((i, k)\) to a given destination \(j\) is as follows:

\[
\tilde{Q}_j^i[i, k](\tau, t) \leq A_j^i[i, k](\tau, t) - S_j^i[i, k](\tau, t)
\]

\[
\leq (1 + \phi_j^i[i, k](\tau, t))\tilde{A}_j^i[i, k](\tau, t) + (1 + \phi_j^n[i, k](\tau, t)) \sum_{n \in NB_i} \tilde{f}_j^n[i, k](\tau, t) - S_j^i[i, k](\tau, t)
\]

(6.12)

The \(\tilde{f}_j^n[i, k](\tau, t)\) is the amount of traffic originated at node \(i\) to destination \(j\) in the interval \((t - \tau)\); the maximum of which is the sum of the tokens in the bucket of the bottleneck link \((i, k)\) and the tokens generated in the interval \((t - \tau)\). In terms of the bucket parameters from the previous section, \(\tilde{f}_j^i[i, k](\tau, t)\) can be written as,

\[
\tilde{f}_j^i[i, k](\tau, t) = w_j^i[i, k](\tau, t) + \tau_j^i[i, k](t - \tau)
\]

(6.13)

Similarly, \(\tilde{f}_j^n[i, k](\tau, t)\) can be obtained as follows:

\[
\tilde{f}_j^n[i, k](\tau, t) = w_j^n[i, k](\tau, t) + \tau_j^n[i, k](t - \tau)
\]

(6.14)

\(w_j^n[i, k](\tau, t)\) and \(\tau_j^n[i, k](t - \tau)\) represents the leaky bucket parameters of bottleneck link \((i, k)\) for the traffic flows originating from node \(n\), \(n \neq i\).
Therefore, substituting for the arrivals and the number of packets serviced in terms of the bucket parameters, Equation 6.12 can be stated as,

\[
\begin{align*}
\bar{Q}_j[i,k](\tau, t) & \leq [(1 + \phi_j[i,k](\tau, t))\{w_j[i,k](\tau, t) + r_j[i,k](t - \tau)\} \\
& + [1 + \phi_j^n[i,k](\tau, t)]\{\sum_{n \in NB_j} [w_j^n[i,k](\tau, t) + r_j^n[i,k](t - \tau)]\} \\
& - r_j[i,k](t - \tau)
\end{align*}
\]  

(6.15)

Since \( \phi_j[i,k] < 1 \) and \( \phi_j^n[i,k] < 1 \), the above equation finally can be written as,

\[
\begin{align*}
\bar{Q}_j[i,k](\tau, t) & < 2w_j[i,k](\tau, t) + r_j[i,k](t - \tau) \\
& + 2 \sum_{n \in NB_j} [w_j^n[i,k](\tau, t) + r_j^n[i,k](t - \tau)]
\end{align*}
\]  

(6.16)

The link delay on link \((i,k)\) can be obtained as,

\[
\begin{align*}
d_j[i,k](\tau, t) & < \delta_j[i,k](\tau, t)\{1 + 2w_j[i,k](\tau, t) + r_j[i,k](t - \tau)\} \\
& + 2 \sum_{n \in NB_j} [w_j^n[i,k](\tau, t) + r_j^n[i,k](t - \tau)]
\end{align*}
\]  

(6.17)

Equation 6.17 gives the approximate bound on the delay on link \((i,k)\). Since link \((i,k)\) is the bottleneck link, its queuing delay must be the longest delay along the equal path link. For the end-to-end delay bound, we assume the \( \theta_j \) is the maximum propagation delay along an equal path link to destination \( j \), and the number of hops from a node to destination \( j \) traversing through link \((i,j)\) is \( K \). Therefore, the end-to-end delay at time \( t \) can be:

\[
d_j(t) < K\theta_j(t)\{1 + 2w_j^f(t) + r_j^f(t) + 2 \sum_{n \in NB_j} [w_j^n(t) + r_j^n(t)]\}
\]  

(6.18)

The proof is done. ◇
6.3.2 Non-negligible Packet Size

In the more general case in which packet sizes are not negligible, the queuing delay or end-to-end delay has to be reconsidered. No cut-through GPS (Generalized Processor Sharing) mentioned in [71] is considered here. That means, no packet is eligible for service until its last bit has arrived. This is a reasonable assumption, because in most network with heterogeneous link speeds, packets are not transmitted until they have completely arrived.

**Theorem 6.3:** When packet size is not negligible, TCMP can achieve an approximate end-to-end delay bound. The delay bound is determined by the maximum propagation delay along the one of equal paths to a destination, the number of hops from source to destination, bucket parameters, and a function of packet size.

**Proof:** Based on the PGPS (Packet GPS) systems, the numbers of packets serviced on the link \((i, k)\) in the period \((t > \tau)\) is given as:

\[
S_j[i, k](\tau, t) + L_i \geq \min_{\tau \in [\tau, t]} \{ [A_j[i, k](\tau, V) - r_j[i, k](\tau, V)] + G_j^m(t - V) \} - m \times L_i \tag{6.19}
\]

where \(L_i\) is the maximum packet size at node \(i\) and \(K\) is the number of hops for a given path from a node to destination \(j\) through link \((i, k)\), \(m = 1, 2, \ldots, K\). \(V\) represents the last time in the interval \([\tau, t]\) that node \(i\) begins a busy period for destination \(j\) and function \(G_j^K\) is a convex function which indicates the amount of service given to destination \(j\) under a greedy regime.

Let \(V_{min}\) be the minimizing value of \(V\). Thus,

\[
S_j[i, k](\tau, t) \geq \min_{V_{min} \in [\tau, t]} \{ [A_j[i, k](\tau, V_{min}) - f_j[i, k](\tau, V_{min})] \}
\]
\[ +G^m_j(t - V_{\text{min}}) - (m + 1) \times L_i \quad (6.20) \]

From the previous session, we can have inequality for the maximum backlog, when \( \tau = 0 \),

\[ \bar{Q}^i_j[i, k](t) \leq 2w^i_j[i, k](t) + 2r^i_j[i, k](t) + 2 \sum_{n \in NB_i} [w^n_j[i, k](t - V_{\text{min}}) + r^n_j[i, k](t - V_{\text{min}})] - G^m_j(t - V_{\text{min}}) + (m + 1) \times L_i \quad (6.21) \]

The link delay of \((i, k)\) can be obtained as:

\[ d^i_j[i, k](t) < \delta^i_j[i, k](t) \{1 + 2w^i_j[i, k](t) + 2r^i_j[i, k](t) + 2 \sum_{n \in NB_i} [w^n_j[i, k](t - V_{\text{min}}) + r^n_j[i, k](t - V_{\text{min}})] - G^m_j(t - V_{\text{min}}) + (m + 1) \times L_i \} \quad (6.22) \]

And the bound on the end-to-end delay is:

\[ d_j(t) < K \theta^j(t) \{1 + 2w^j(t) + 2r^j(t) + 2 \sum_{n \in NB_i} [w^n_j(t - V_{\text{min}}) + r^n_j(t - V_{\text{min}})] - G^m_j(t - V_{\text{min}}) + (m + 1) \times L_i \} \quad (6.23) \]

The above inequality shows that the end-to-end delay depends on the bottleneck bandwidth as before. In addition, it is also a function of the packet size. ∎
CHAPTER 7

A SIMULATION STUDY

This chapter presents simulation results. The simulation began in the early stages of the research work. In the process of simulation, a lot of complicated network situations were clearly understood, and many unpredictable network parameters were measured.

First, using simple network topologies, simulation is used to compare the results of three routing methods, i.e. CMR, RMR and SPR, and identify the factors that affect their performance. These routing methods are theoretically analyzed in Chapter 4. Following the comparison, an extensive series of experiments are conducted to evaluate the performance of the TCMP scheme described in the previous chapter. The results of these experiments are presented based on three key performance parameters: the delay encountered when new flows are initiated, the total packet losses, and link utilization.

7.1 Simulator Design

To closely capture the Internet characteristics, the simulator does packet-level simulation of packet transmission and routing information distribution. The simulator uses link-state routing illustrated in the previous chapter. The ns network simulator [69] is extended to be suitable to our simulation. The simulation utilizes some functions of ns. For example, we adopted its network topology generation function to build topologies used in our simulations, its traffic generator to generate real-time traffic, and its trace and monitoring support to monitor packet queuing, packet drops,
and link utilization. In *ns*, static and session routing use the Dijkstra’s SPF algorithm (link-state routing algorithm), and its dynamic routing uses the distributed Bellman-Ford algorithm (distance-vector routing algorithm). Furthermore, *ns* dynamic unicast routing does not calculate route based on network statistics, it only changes its route when network topology has changed.

We extend the *ns* network simulator in the following parts:

- Unicast routing uses distributed link-state algorithm, and a router can dynamically change its routing table based on link load.

- Unicast routing uses two-level routing table: the first level is built based on link propagation delay, and the second level is built based on the available bandwidth.

- Unicast routing supports class-based multipath routing.

- Packet forwarding function classifies the packets of a traffic according to the class of traffic flow.

- Leaky bucket mechanism is implemented in each router to monitor link load on each of the multiple paths.

### 7.1.1 Traffic Load

Traffic used in simulation belongs to a Constant Bit Rate (CBR) model which uses an ON/OFF model with exponentially distributed ON and OFF times. In the ON state, a source produces a (exponentially distributed) number of data packets with some constant inter-packet generation time, which is determined by the peak rate. The source then stays idle for an exponentially distributed duration before starting the transmission of the next train of packets. The traffic model is shown in Figure 7.1.
In most cases, the packet average burst time is set to be 100ms; average idle-
time is 50ms; peak rate is 100kbps; and packet size is 210 bytes. It is noted that the 
exponentially distributed ON/OFF model does not exactly model the real-time traffic. 
The holding time distribution of most real-time applications, such as conversations, 
facsimile, and voice mail, has a large portion of very short calls and lognormal long-
tail distributions. Since the simulation is focused on comparing the performance of 
routing algorithms, the simple exponentially distributed ON/OFF traffic model is 
suitable.

7.1.2 Performance Metrics

The performance metric for traditional data application is the average network 
throughput for the best-effort traffic. The average throughput is defined as:

\[
\text{Average throughput} = \frac{\text{bytes received at destinations}}{\text{the measurement interval}}
\]

Average throughput is a suitable performance metric for measuring the best-effort 
traffic transmitted by TCP protocol. However, for traffic flows transmitted by UDP 
protocol, end-to-end delay, packet drops, and call blocking rate are suitable perfor­
ance metrics. The end-to-end delay measures the period that a packet traverses 
from a source to a destination. The packet drops measure the packet losses at the
routers when a traffic flow is being transmitted along a path from a source to a destination. The call blocking rate is for measuring guaranteed traffic transmission which needs to be supported by admission control and resource reservation. The call blocking rate is defined as: the percentage of sessions being rejected by the network over the total number of arrival sessions, e.g.

\[
\text{Call blocking rate} = \frac{\text{number of rejected guaranteed sessions}}{\text{number of arrival guaranteed sessions}}
\]

A guaranteed session can be rejected either because no path with sufficient resources can be found by the routing algorithm or because the resource availability on the selected path has changed since the time when the routing decision was made.

7.1.3 Updating Mechanism

Each node measures the link load of its outgoing links at each sample period. The sample time is set to be 0.1 second. Three types of methods are considered to update network information which are:

1. each node triggers an update in some specific period \( t \) seconds. The update interval \( t \) is uniformly distributed between 0.9\( t \) to 1.1\( t \) seconds.

2. if the measured load is larger than 90 percent of the link's capacity, the node initiates an update to indicate the current load of its adjacent link.

3. if packet drops in a node are more than some fixed amounts, the node initiates an update to indicate the current load of its adjacent link.

We do not adopt the actual flooding protocol in the simulator to transfer update information. In our simulation, disseminating the network load is the operation that delivers a copy of an update directly to each node. This dissemination method reduces the number of packet level events that are simulated; consequently it reduces the
simulation running time. The disadvantage is that it introduces a small error into the simulation. Since we focus on the comparison of various routing algorithms instead of the actual performance measurement for routing algorithms, the error does not affect the comparison results.

7.2 The Comparison of Class-based Routing with Other Routing Schemes

The simulation models for a 6-node networks are employed as shown in Figure 7.2 and Figure 7.3. The differences between these two network topologies are in link parameters and connectivity degrees. The higher the connectivity degree, the more multiple paths are produced. We use these small size networks because results are easier to understand and explain. The numbers marked on each link in the two networks are link’s available bandwidth and propagation delay. We set the buffer size for each node at 50 packets.
7.2.1 Static Routing

Figure 7.2 is used to compare the end-to-end delay that a flow experiences when there is another flow joining network during its transmission.

From Figure 7.2, we can see that there are two paths that have equal propagation delay from node $n_0$ to node $n_5$. The two paths are $n_0-n_1-n_3-n_5$ and $n_0-n_2-n_4-n_5$. The bottleneck bandwidth for the equal paths are 100kb and 64kb, which are in $n_1-n_3$ and $n_2-n_4$, respectively. There are two traffic flows from source node $n_0$ to destination $n_5$. Flow 1 starts to transmit at $n_0$ at 0.1 second, and flow 2 starts at 3.0 second. When the SPR method is used, two flows travel the same path: $n_0-n_1-n_3-n_5$. In the CMR method, flow 1 is transferred by path $n_0-n_2-n_4-n_5$, and 3 seconds later, flow 2 comes at $n_0$ and is routed to $n_0-n_1-n_3-n_5$. In the RMR method, two flows are transferred randomly to each of the two paths. The simulation runs for 20 seconds. The packet drops and queueing delays are measured from link $n_1-n_3$ and link $n_2-n_4$. Figure 7.4 and 7.5 show the results.

There are no packet drops for the CMR method, and the SPR experiences high
Figure 7.4: Packet Drops with Running Time

Figure 7.5: Queueing Delay with Running Time

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
packet drops after 3 seconds, e.g. after the second flow started at $n_0$. In Figure 7.5, the CMR method has a queueing delay at the beginning. This is because the flow has a 100 kbps peak rate and the path has a 65 kbps bottleneck bandwidth. There are 35 kbps (around 27 packets) needed to be buffered during bursty period, and the flow experiences a 0.7 second queueing delay. The important thing is that the queueing delay is stable after another flow starts to be transmitted, because the second flow traverses another path. On the contrary, the RMR and SPR experience queueing delays after 3.0 seconds. The increased delay produces high delay jitter for flow 1. This is the main problem which needs to be solved for a real-time application. It is noted that the SPR has less delay than the RMR. The reason is that SPR drops the extra packets when they are waiting in the buffer.

7.2.2 Dynamic Routing

When a dynamic routing is implemented, the update period has to be carefully set, otherwise, it may not always reduce packet losses and increase network throughput. Figure 7.3 is used as a network topology for simulation. All the link bandwidth is 1Mbit/sec, and the link propagation delay is 1ms. Traffic flows are created between source-destination pairs as shown in Table 7.1. Since simulation is used to test the delay and packet drops for flows from node $n_0$ to $n_5$, $n_0$ is treated as the root node and $n_5$ is treated as the leaf node. The source-destination pairs only consider nodes passed by flows from root to leaf node. At the initial time, the equal paths between each source-destination pairs in Table 7.1 are shown in Table 7.2.

The simulation first tests routing dynamic behavior. The traffic load is configured in evenly distributed manner. This means that all source-destination pairs trigger almost the same amount of traffic sessions. From each source-destination pair, traffic flows start to transmit with a uniform distribution between 0 to 10.0 seconds. Each

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
<table>
<thead>
<tr>
<th>source-destination pairs</th>
<th>one hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0,1),(0,2)</td>
<td></td>
</tr>
<tr>
<td>(1,2),(1,3),(1,4)</td>
<td></td>
</tr>
<tr>
<td>(2,1),(2,3),(2,4)</td>
<td></td>
</tr>
<tr>
<td>(3,2),(3,4),(3,5)</td>
<td></td>
</tr>
<tr>
<td>(4,1),(4,3),(4,5)</td>
<td></td>
</tr>
<tr>
<td>two hops</td>
<td></td>
</tr>
<tr>
<td>(0,3),(0,4)</td>
<td></td>
</tr>
<tr>
<td>(1,5)</td>
<td></td>
</tr>
<tr>
<td>(2,5)</td>
<td></td>
</tr>
<tr>
<td>three hops</td>
<td></td>
</tr>
<tr>
<td>(0,5)</td>
<td></td>
</tr>
</tbody>
</table>

Table 7.1: Source-Destination Pairs of Traffic Flows

<table>
<thead>
<tr>
<th>src-dst pairs</th>
<th>equal paths</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0,3)</td>
<td>(0-1-3),(0-2-3)</td>
</tr>
<tr>
<td>(0,4)</td>
<td>(0-1-4),(0-2-4)</td>
</tr>
<tr>
<td>(1,5)</td>
<td>(1-3-5),(1-4-5)</td>
</tr>
<tr>
<td>(2,5)</td>
<td>(2-3-5),(2-4-5)</td>
</tr>
<tr>
<td>(0,5)</td>
<td>(0-1-3-5)</td>
</tr>
<tr>
<td></td>
<td>(0-1-4-5)</td>
</tr>
<tr>
<td></td>
<td>(0-2-3-5)</td>
</tr>
<tr>
<td></td>
<td>(0-2-4-5)</td>
</tr>
</tbody>
</table>

Table 7.2: Equal Paths from Source-Destination Pairs
traffic flow lasts 5.0 seconds. Totally there are seven traffic flows starting at each node during the period of 0 to 10.0 seconds.

The system runs for 20 seconds. The first update method is adopted, and the update period can be tuned from the first second to the 11th second. It is found that the packet drops and link utility are changed with the update interval as shown in Figures 7.6 and 7.7.

The serious packet drops and high link utility is at the update period between the 6th second to the 8th second, and between the first to 3rd second. The reason is explained as follows:

- The update information is the highest priority traffic in a network. At the update time, if all nodes in a network immediately start to disseminate the state information to their neighbors, the network would be totally loaded by the highest priority traffic. It will affect the normal traffic transmission. Thus, the updating time at each node is generally uniformly distributed around the fixed

Figure 7.6: Packet Drops with the Increasing of Updating Interval
Figure 7.7: The Times of Link Utility Over 90 Percent of its Original with Increasing Update Interval

period. Since nodes change their routing tables asynchronously, a temporary routing loop may be created. Therefore, during each node's changing its routing table, traffic flow may not be routed to the correct next hop, and packet drops are produced.

- In the period between the 6th to the 8th, there are more traffic flows in the system compared with other period. Figure 7.8 gives the flow distribution in the system. Every flow lasts 5.0 seconds in the system.

- Even though the update interval is between the first to the 3rd second, there are still high packet drops and high link utility. The reason is that a frequently updating routing table may affect a packet's transfer even under a light link load.

In the real network environment, this situation is true. More frequent updating
makes the network oscillate more often and may cause routing loops; furthermore, it produces more packet drops, especially when the network is heavily loaded.

According to the above analysis of routing dynamic behavior, the next simulation sets the update interval at 9 seconds, and compares the performance of three routing schemes. The simulation is still running for 20 seconds. There are two types of traffic load: background load and focused load. The background load is used to simulate traffic routine in a network. The focused load is used to measure the performance when more traffic is initiated in the network. The measured performance metrics are packets in queue, packet drops and link utilization. The background traffic flows are created uniformly in each source-destination pair between 0 to the 10th second, and each flow lasts for 5 seconds. During the running period of the background load, there are focused load initiated from node $n_0$ to destination node $n_5$. The focused load contains three traffic flows, which have different classes, i.e, class 1, class 2 and class 3. The three flows, each has a peak rate 150kbps, 200ms burst-time, and 50ms
idle-time. They have the same exponential ON/OFF distribution as background load. The durations of the three classes of flows are 4.0th to 8.0th second, 2.0th to 6.0th second, and 3.0th to 7.0th second, respectively. In Figure 7.9, it is noted that SPR experiences more packet drops when each node triggers more than 4 sessions of traffic flow. The CMR and RMR are almost the same for the packet drops, which is produced after flow is triggered by each node more than 9 sessions. Therefore, the following comparison only considers the packet delay by using the RMR and CMR methods. Since the number of packets in the buffer decides the queuing delay, the simulation only measures the number of packets in the buffer instead of measuring the time. Figures 7.10 and 7.11 show the packets in the buffer when each node triggers 7 and 8 sessions of flow, respectively. It can be seen that during the period of the three new flows transmitting, the packets in the system experience a high queuing delay in both situations. However, the RMP produces higher queuing delay than the CMR.
Figure 7.10: Packet in Queue when Each Node Triggers 7 Sessions of Flow and Update Interval is 9 Seconds

Figure 7.11: Packet in Queue when Each Node Triggers 8 Sessions of Flow and Update Interval is 9 Seconds
Figure 7.12: Packet Drops with Increasing Flows in Each Source-Destination Pair when Update is Based on Link Utilization

Figure 7.13: Updates with Increasing Flows in Each Source-Destination Pair when Update is Based on Link Utilization
In the last simulation of this section, the update is based on the link load. When the link load is over 90 percent of its capacity, it triggers an updating routine. From Figure 7.12 and Figure 7.13, we note that the results are similar to the results when the update is based on the fixed updating interval. The SPR still has the worst performance among the three routing algorithms. A traffic experiences a lot of packet drops when the SPR is used. The SPR also produces more link updates, so it is more unstable than the other two methods. Comparing the RMR and CMR methods, they have similar performance for packet drops. When the background load is light, e.g. less than 7 sessions, the CMR has lower link updates than the RMR. However, when the background load is heavy, the RMR is more stable and produces less updates than the CMR algorithm.

7.3 Traffic Performance Using TCMP Scheme in a Small Network

In this section, we examine the performance of the TCMP scheme described in Chapter 6. We simulate two situations for the TCMP scheme: one is called Static Bucket TCMP (TCMP-SB); another is called Dynamic Bucket TCMP (TCMP-DB). In TCMP-SB algorithm, a leaky bucket for each equal path does not change its bucket size during network stable period. In TCMP-DB algorithm, on the other hand, the bucket size is dynamically determined by the traffic load forwarded to the path which the bucket is monitoring. Actually, when the bucket sampling interval in the TCMP-DB algorithm is tuned to large value, the TCMP-DB becomes TCMP-SB.

The network topology used in this section is Figure 7.3. We consider both scenarios in which the network load is evenly and unevenly distributed.
7.3.1 Evenly Distributed Load in Quasi-Static Routing

A quasi-static routing means that the routing algorithm does not change its routing table so often. All routing algorithms evaluated in this section use the 3rd update method, e.g. updating their routing table according to packet drops. When the packet drops at a node is more than 100 packets, the node initiates an update. Actually, this update condition makes the network very stable in most cases. When packet drops are more than 100 packets, the network is heavily loaded.

The simulation runs for 30 seconds. Traffic flows are evenly created in each source-destination pair between 0 to the 25th second. The source-destination pairs are the same as in Section 7.2.3, as shown in Table 7.1. Each of the flows lasts 5.0 seconds. During the background load, focused load is initiated from node $n_0$ to $n_5$. The focused load has a total of six flows which are classified into three class, i.e. class 1, class 2 and class 3. The start time for the flows are such that class 1 is 0.5 and 0.6 second for two flows; class 2 is 1.0 and 1.1 seconds for two flows; class 3 is 1.5 and 1.6 seconds for the remaining two flows.

From Figure 7.14, we see that the performance variation between the different routing algorithms can be large. The RMR algorithm performs better than other three algorithms and the SPR algorithm performs the worst. When each source-destination pair has 13 sessions of traffic, the SPR starts to have high packet drops, or a total of 234 packet drops; the other algorithms have zero packet drops. At 15 sessions of traffic, the TCMP-SB experiences 40 packet drops, and the SPR has up to 1114 packet drops. It is noted that the TCMP-SB and TCMP-DB algorithms perform much better than the SPR, and the TCMP-DB has a performance similar to the RMR algorithm. The reason for the result is that the load is evenly distributed, and the RMR algorithm also transfers traffic evenly among equal path. On the other hand, the SPR always tries to transfer all traffic along a single path. The TCMP-SB
algorithm does not monitor the link load; it cannot change the traffic forwarding based on the link state. The TCMP-DB algorithm tries to balance link load dynamically; however, it still needs some time to adjust the packet forwarding, so it can not be better than the RMR algorithm.

7.3.2 Unevenly Distributed Load in Quasi-Static Routing

We still adopt quasi-static routing here, and use the same updating mechanism. Even though networks are typically designed to match the expected traffic conditions, the network load can often unevenly be distributed in the sense that the traffic load does not precisely match the expected load, resulting in higher loads in some parts of the network than in others. We simulate such scenario with unevenly distributed load by having some source-destination pairs transferring more sessions than others. Table 7.3 shows the unevenly loaded source-destination pairs. Lightly loaded pairs
have a total of 10 sessions of traffic to be transmitted. The simulation measures the packet drops when the heavily loaded pairs change their traffic load from 5 sessions to 20 sessions. The bucket sampling interval is set to be 1.0 second.

Figure 7.15 shows the packet drops under unevenly loaded traffic. We observe similar behavior for the SPR routing algorithm as for evenly loaded situation. When heavily loaded links transmit 15 sessions, the SPR starts to experience high packet drops: up to 300 packet drops. However, the other algorithms have zero packet drops. The TCMP-DB performance is the best. The performance for the RMR is between the TCMP-SB and the TCMP-DB. The reason is that the RMR algorithm is not suitable to the unevenly loaded network. The TCMP-DB can perform better than the other algorithm, since it can adjust the traffic forwarding according to the link load.

By comparing Figures 7.14 and 7.15, it is noted that the SPR algorithm performs better in unevenly loaded network than evenly loaded network. The reason is that the SPR always chooses the widest path (e.g. high bandwidth path) among the equal paths to transfer traffic flows.

<table>
<thead>
<tr>
<th>lightly loaded links</th>
<th>heavily loaded links</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0-1)</td>
<td>(0-2)</td>
</tr>
<tr>
<td>(1-3)</td>
<td>(2-3)</td>
</tr>
<tr>
<td>(3-5)</td>
<td>(2-4)</td>
</tr>
<tr>
<td>(1-4)</td>
<td>(4-5)</td>
</tr>
<tr>
<td></td>
<td>(0-1-3)</td>
</tr>
<tr>
<td></td>
<td>(0-2-3)</td>
</tr>
<tr>
<td></td>
<td>(0-1-4)</td>
</tr>
<tr>
<td></td>
<td>(0-2-4)</td>
</tr>
<tr>
<td></td>
<td>(1-3-5)</td>
</tr>
<tr>
<td></td>
<td>(1-4-5)</td>
</tr>
<tr>
<td></td>
<td>(2-3-5)</td>
</tr>
<tr>
<td></td>
<td>(2-4-5)</td>
</tr>
</tbody>
</table>

Table 7.3: Unevenly Loaded Links
7.3.3 Unevenly Distributed Load in Dynamic Network

This section evaluates the dynamic behavior of the algorithms. The updating is still based on packet drops; however, a node initiates an update when it has 10 packet drops. Therefore, the network is more dynamic than quasi-static routing. We make link load further unevenly loaded, as shown in the Table 7.4. The lightly loaded links have 8 sessions of traffic and the heavily loaded links have 40 sessions of traffic. The simulation measures the packet drops according to the bucket sampling interval. From Figure 7.16, it is noted that the TCMP-SB performs better than other algorithms. It is interesting to see that when the bucket sampling interval is short, the TCMP-DB performs the worst, however, when the bucket interval becomes long, it converges to that of the TCMP-SB. The SPR outperforms the RMR and TCMP-DB when the bucket sampling interval is less than 12 seconds. This is true since the SPR is more suitable for use in unevenly loaded network than in evenly loaded network.
Table 7.4: More Unevenly Loaded Links

<table>
<thead>
<tr>
<th>lightly loaded links</th>
<th>heavily loaded links</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0-1)</td>
<td>(0-2)</td>
</tr>
<tr>
<td>(1-3)</td>
<td>(2-3)</td>
</tr>
<tr>
<td>(3-5)</td>
<td>(2-4)</td>
</tr>
<tr>
<td>(1-4)</td>
<td>(4-5)</td>
</tr>
<tr>
<td>(0-1-3)</td>
<td></td>
</tr>
<tr>
<td>(0-2-3)</td>
<td></td>
</tr>
<tr>
<td>(0-1-4)</td>
<td></td>
</tr>
<tr>
<td>(0-2-4)</td>
<td></td>
</tr>
<tr>
<td>(1-3-5)</td>
<td></td>
</tr>
<tr>
<td>(1-4-5)</td>
<td></td>
</tr>
<tr>
<td>(2-3-5)</td>
<td></td>
</tr>
<tr>
<td>(2-4-5)</td>
<td></td>
</tr>
</tbody>
</table>

Figure 7.16: Packet Drops with Increasing Flows in Each Heavily Loaded Source-Destination Pair in Heavily Unevenly Load Situation
From Figures 7.15 and 7.16, we can see that the TCMP scheme generally performs better than the SPR and RMR schemes in the most situations. The TCMP-DB is more suitable to a network using static routing; on the other hand, the TCMP-SB performs well in dynamic routing.

There are two parameters for deciding the situation of TCMP-DB and TCMP-SB, which are bucket sampling interval and the difference of bucket size among the equal paths. When the bucket sampling interval is to be a long period, TCMP-DB is similar to TCMP-SB. When comparing the bucket sizes of equal paths, the difference of their bucket size decides when the traffic needs to be forwarded to other path. If the difference is set to a big value, the TCMP-DB is similar to TCMP-SB; otherwise, TCMP-DB has more dynamic behavior.

7.4 Traffic Performance Using TCMP Scheme in a Large Network

In this section, we compare the routing performance using a large network topology. The MCI network is adopted as the topology in our simulation. The real topology is shown in Figure 7.17. $T_3$ line has 45 Mbps bandwidth and $OC_3$ is at 155 Mbps by using optical carrier. The parallel series lines between two nodes are used to increase the network reliability. They can not be modeled in the simulator, so we add extra node between one of the parallel lines, and configure two lines as equal paths. The topology used in the simulation is shown in Figure 7.18.

Since the high bandwidth in the MCI link increases the simulation time, we configure the link bandwidth in the simulation topology to be 100 times less than the original value. The link bandwidth in Figure 7.18 is 0.45 Mbps for $T_3$ link and 1.55 Mbps for $OC_3$ link. Propagation delay for $T_3$ link is 10 msec, and $OC_3$ is 100 msec without considering the physical distance in each link.

We focus on unevenly distributed load network, since it is more like the practical
situation. We simulate the unevenly distributed load situation by having a percentage of the sessions selected as a source and destination pair from a preselected subset of the nodes, while the rest of the sessions pick any node as their source and destination. The exponentially distributed ON/OFF model is still used for modeling traffic flows in this section. The traffic average burst time is still set to 100 msec; idle-time is 50 msec; peak rate is 100 kbps; packet size is 210 bytes.

The simulation still uses dynamic routing. The update function is triggered when there are more than 10 packet drops. The bucket sampling interval is set to be 2.0 seconds. The simulation runs for 30 seconds. Traffic sessions start uniformly from 0 to 25 seconds, and they all have 5.0 second transmitting duration. We configure the traffic flows with 100 sessions for the lighter traffic load and more than 100 sessions for the heavier traffic load.

Figure 7.19 shows the packet drops for different routing algorithms when the heav-
Figure 7.18: MCI Topology Used in the Simulations

Network load changes from 100 sessions to 300 sessions. From the figure, we can see that the SPR performs the worst. The other three algorithms have similar performance when the heavier load is less than 250 sessions. The TCMP-DB's performance is somewhat better than those of TCMP-SB and RMR algorithms when heavier load is less than 220 sessions. When the heavier load is more than 250 sessions, the TCMP-SB outperforms the TCMP-DB and RMR algorithms. This result is similar to Figure 7.15, which we obtained from the simple topology, and the insight result is that when the network has serious unevenly distributed load, dynamically adjusting traffic forwarding cannot be more advantageous than that just forwarding traffic to the same path in the stable period.

Figure 7.20 gives the packet drops when the bucket sampling interval changes. The traffic load is set to be 100 sessions for the lighter load and 300 sessions for the heavier load. The performance for TCMP-DB, TCMP-SB and RMR algorithms is similar to
Figure 7.19: Packet Drops with Heavier Sessions in Uneven Loaded Situation

Figure 7.16 for simple topology. The TCMP-SB performs better than others. The TCMP-DB converges to the TCMP-SB when the bucket sampling interval is 8.0 seconds. The RMR is better than the SPR. The SPR performs the worst. The reasons for difference in performance for the SPR in the simple and MCI topology is two-fold. One is the traffic load which is different in two topologies. The other is that the SPR performance tends to be sensitive to network topology.

7.5 Supporting Guaranteed Services

First, we still use Figure 7.3 topology to evaluate TCMP performance for supporting resource reservation. If multipath exists between source and destination, the original method chooses one of multiple paths randomly for resource reservation. On the other hand, the TCMP chooses a path with largest bucket size from the equal paths.

The traffic load is still ON/OFF model with exponentially distributed ON and
Figure 7.20: Packet Drops with Bucket Sampling Interval in Dynamic Situation

OFF times. The configuration of background traffic is based on Table 7.1. The heavier loaded links belong to (0,1), (1,3), (3,5), (1,4) source-destination pairs, and lighter loaded links belong to other source-destinations. The simulation uses quasi-static routing with update being triggered when there are more than 100 packet drops. The bucket sampling interval is set to be 3.0 seconds.

The simulation runs for 30 seconds. The heavier or lighter sessions are initiated from each source-destination pairs between 0 to 25 seconds with uniform distribution. The lighter load has 20 sessions, and heavier load changes from 25 to 42 sessions. The real-time traffic which needs guaranteed service has the same bandwidth requirement for 2.0 kbps. The calling interval, e.g. the interval for the traffic to ask resource reservation, is 0.5 second. There is no holding time for each traffic, that means once they reserve the bandwidth, the bandwidth is held until the end of the simulation. There are totally 20 calls or sessions for guaranteed service, and they are initiated from node 0 and ended at node 5.
Figure 7.21 shows the comparison result. The original reservation method experiences 0.2 call blocking rate when heavier loaded links have 25 sessions of traffic; however, TCMP-DB allows all calls to get into the network. TCMP-DB can generally obtain low call blocking rate compared to the original reservation method without the supporting of routing information.

We further evaluate TCMP performance in the MCI topology in Figure 7.20. The network parameters and unevenly loaded traffic are configured the same as that in Section 7.4. The simulation uses quasi-static routing. The update function is triggered when there are more than 100 packet drops. The bucket sampling interval is set to be 3.0 seconds. The simulation still runs for 30 seconds.

The heavier load links initiate 300 sessions of traffic flows, and the lighter load links initiate 100 sessions of traffic. Figure 7.22 shows the call blocking rate when the number of calls is increased. The original method starts to reject calls when there are
Figure 7.22: Call Blocks with Increasing Call Number in Unevenly Loaded Situation

more than 30 calls; on the other hand, TCMP-DB rejects calls when the number of calls is more than 35. It is clear that knowing the bottleneck bandwidth is beneficial for guiding the resource reservation.
CHAPTER 8

CONCLUSIONS AND FUTURE WORK

In this dissertation, we have examined QoS-based routing in the Internet environment. One of the basic problems we tackled was how to characterize network resources for implementing QoS-based routing. The detailed analysis of QoS requirements and the goal of QoS-based routing provide a clear picture for deciding routing metrics. Furthermore, other research on QoS-based routing has also inspired us to develop a suitable scheme to construct routing tables.

The second basic problem we considered was what granularity of routing can satisfy QoS-based routing requirements. Through theoretical analysis and evaluation by simulation, we decided that our routing architecture utilizes a class-based routing scheme.

The final basic problem we considered was how to route traffic flows according to their different kind of service requirements. To route traffic efficiently, i.e. satisfying traffic's delay and loss probability requirements, a router needs to have up-to-date information about network available resources. In a network, one of the main difficulties that the routing faces is delayed feedback. The delay in the feedback information poses a fundamental limit to any feedback control mechanism. Any attempt to adjust a routing decision faster than the speed that the information can propagate only results in wild oscillation. On the other hand, to keep network information up-to-date, flooding updated information is essential. The frequent flooding of routing information consumes valuable network resources, i.e. link bandwidth and router's processing time. The solution that we proposed to solve the problem was trying to utilize a mon-
itor at each router for monitoring traffic forwarding. The combination of monitoring and out-of-date information is the basis of controlling traffic forwarding.

This dissertation has made a number of contributions to the area of QoS-based routing, which are summarized in the following section. Further, we conclude the dissertation by suggesting several avenues for future work based on this research.

8.1 Summary of the Contributions

1. A detailed discussion on the characteristics of the QoS-base routing algorithm is presented. The limitations of current Internet routing architecture for multimedia applications are analyzed.

2. Through theoretical analysis and simulation, the behavior of class-based routing is examined by comparing it with single-path and random multipath routing algorithms. The simulation can help one to understand the dynamic behavior of various routing methods. The results show that class-based routing provides an efficient method for routing to satisfy end-to-end delay bound for real-time traffic.

3. A new framework for QoS routing is presented. The new algorithm is called Two-Level Class-based Multipath Routing with Prediction (TCMP). This algorithm differs from existing routing protocols used in Internet routing providing best-effort service in the following important ways:

   - Dynamic distributed multipath routing is utilized.
   - Multipath routing is computed between source and destination to enable increased network throughput. The routing includes the first-level metric based on link propagation delay, and the second-level metric based on the bottleneck bandwidth.
• Leaky buckets are used as guidance for the bottleneck bandwidth of each equal path to control packet forwarding at each node and further support resource reservation.

4. The approximate worst-case bounds on the bottleneck delay and end-to-end delay are derived according to the TCMP algorithm.

5. The TCMP algorithm is evaluated using simulation in a variety of network conditions. The simulation results show that the TCMP can reduce packet drops and increase network throughput in any size network topology.

6. By using the monitoring mechanism in TCMP, we can also improve the scalability of QoS-based routing. The monitoring mechanism can limit distribution of dynamic information; therefore, it can reduce routing overhead.

This research is significant because it provides a comprehensive examination of how routing can best support real-time traffic. In addition to describing a framework for routing in this environment, this research provides an understanding of factors that affect routing performance. An understanding of these issues will be useful to network architects who are working to make real-time service a reality in packet-switched networks.

8.2 Future Work

Possible future directions of this research are as follows:

A. The algorithm proposed in this dissertation is directly applicable to the Internet. We believe that the algorithm can greatly enhance QoS-based routing over the Internet. Due to resource constraints, experiments were carried out by simulation. Implementing the algorithm into the Internet testbed and carrying out real experiments would be beneficial.
B. All of our discussions and evaluations assumed networks with no hierarchy. However, today’s Internet makes use of hierarchy and distinguishes between intra-domain and inter-domain routing. How to adapt the TCMP to fit this architecture would be an interesting future work.

C. Applying the concepts presented to multicast routing protocols requires additional research. We confined our current investigation to unicast routing, even though some real-time applications will often require support from multicast routing. Current multicast routing protocols will have the same problem supporting real-time multicast traffic as today’s unicast protocols will have with unicast real-time traffic. Multicast routing tables are computed using static metrics and they are only updated in response to changes in network topology. New multicast protocol has been proposed to address the multicasting scalability problem [33]. We believe the concepts we proposed for unicast routing can be applied to the multicast routing for supporting real-time applications.

D. Efficiently Integrating our routing algorithm with other components of a real-time architecture, such as admission control and scheduling algorithms, is very important. In future high speed networks, the integration of routing, admission control and scheduling will form an ultimate resource control system for the network which provides integrated traffic control and resource management. In such a system, it is important to have a unified information database and effective mechanism for message passing between different components. It is also desirable to have distributed control with a certain degree of global coordination [91].

This research has addressed one aspect of the problem of providing real-time service in Internet. While this dissertation outlined and evaluated a solution to the problem of routing in this environment, a number of issues clearly remain unanswered.
We believe that this research will contribute to an understanding of the issues relevant to the realization of real-time service in packet-switched networks. We also hope that it will stimulate additional interest in the important research area of real-time service in general, and in network routing in particular.
BIBLIOGRAPHY


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.


VITA

Graduate College
University of Nevada, Las Vegas

Bing Chen

Local Address:
4214 Claymont St. #4
Las Vegas, NV 89119

Home Address:
1522 Ambergrove Dr.
San Jose, CA 95131

Degrees:
Bachelor of Engineering, Electrical Engineering, 1987
Liaoning University of Industry, P.R.China

Master of Science, Electrical Engineering, 1990
Shenyang Institute of Automation,
Chinese Academy of Science, P.R.China

Special Honors and Awards:
Honored as an outstanding young scholar in Science and Engineering
by UNLV Woman in Science and Engineering, 1998

Publications:
“Effective Deployment of ATM into PACS”,

“Two-level Routing for Multimedia Applications”, the 11th International
Conference on Computer Applications in Industry and Engineering,

“Analysis of QoS-based Routing in Large Packet Switched Networks”,
IEEE Symposium on Performance Evaluation of Computer and
Dissertation Title: QoS-based Multipath Routing for the Internet

Dissertation Examination Committee:
Chairperson, Dr. Shahram Latifi, Ph.D.
Committee Member, Dr. Lori M. Bruce, Ph.D.
Committee Member, Dr. Eugene McGaugh, Ph.D.
Committee Member, Dr. Ajoy K. Datta, Ph.D.
Graduate Faculty Representative, Dr. Kazem Taghva, Ph.D.