ABSTRACT

QOS ROUTING
SOURCE ROUTING PROBLEMS AND SOLUTIONS

by
Ye Tian

The notion of Quality-of-Service has been proposed to capture qualitatively or quantitatively defined performance contracts between the service provider and the user applications. Integrated network services are designed to support Quality-of-Service (QoS). One of the primary goals for the integrated network services is to find the paths that satisfy given QoS requirements, namely QoS routing. The challenging issue in this area is to route packets subjected to multiple uncorrelated constraints because the problem is inherently NP-complete. This thesis studies the source routing heuristic approaches that bring the time complexity down to the polynomial-time for the multi-constrained path (MCP) problem. A new source routing framework (SRDE) is further proposed to tackle this problem. The theoretical analysis and simulation results demonstrate that the proposed framework is capable of integrating existing source routing algorithms, resulting in better performance in terms of the time complexity and success ratio.

QOS ROUTING
SOURCE ROUTING PROBLEMS AND SOLUTIONS

by
Ye Tian

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Electrical Engineering

Department of Electrical and Computer Engineering

January 2002

APPROVAL PAGE

QOS ROUTING
SOURCE ROUTING PROBLEMS AND SOLUTIONS

Ye Tian

12/20/2001

Dr. Nirwan Ansari **Date**
Professor of Electrical and Computer Engineering, NJIT

12/20/2001

Dr. Lev Zakrevski **Date**
Assistant Professor of Electrical and Computer Engineering, NJIT

12/20/2001

Dr. Edwin Hou **Date**
Associate Professor of Electrical and Computer Engineering, NJIT

BIOGRAPHICAL SKETCH

Author:	Ye Tian
Degree:	Master of Science in Electrical Engineering
Date:	January 2002

Undergraduate and Graduate Education:

- Master of Science in Electrical Engineering
  New Jersey Institute of Technology, Newark, NJ, 2001

- Bachelor of Science in Electrical and Electronic Engineering
  University of Canterbury, Christchurch, New Zealand, 1999

- Bachelor of Science in Automation
  Tsinghua University, Beijing, P. R. China, 1995

Major:	Electrical Engineering

To my beloved family

ACKNOWLEDGMENT

I wish to thank Prof. Ansari and Prof. Zakrevski for the fruitful discussions that I have had with them during the entire period of my master studies. Thanks also go to the committee member Prof. Edwin Hou for his attendance for our routine meetings and valuable suggestions.

I have been using the facilities in Advanced Networking Laboratory during my master studies. I appreciate all the lab members for their co-operation and want to thank them all for their great assistance to my work.

I also want to thank the donor of Phonetel Fellowship for supporting me through my master program. Most importantly, it is such a great honor and has been encouraging me all the time.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The success of the Internet infuses new challenges to networks. Standard Internet Protocol (IP)–based network supports best-effort data delivery by default, which has provided satisfactory services for various applications, such as the email and file transfer, to a great extent. The notion Quality-of-Service (QoS) has been proposed to capture the qualitatively or quantitatively defined performance contracts between the service provider and the user applications [1]. Best-effort IP allows the complexity to stay within the end-hosts, so the network can remain relatively simple. However, the network is in absence of QoS measures. Sufficient bandwidth seems to be a good remedy for compromised performance of the network due to congestions and delays. Nevertheless, over-providing bandwidth wastes valuable network resources. More or less, it ends up with guaranteeing certain traffics while punishing some others. Especially for the real-time multimedia applications, the degradation can be severe as a result of insufficient network resources. For example, delays cause problems for the real-time multimedia delivery and insufficient bandwidth compromises the video quality.

1.1 QoS in ATM

Contrary to the Internet, Asynchronous Transfer Mode (ATM) is one of the integrated network services that are designed to provide Quality-of-Service. ATM is a switched technology, which implies dedicated bandwidth. ATM uses fixed transmission units, or cells, which allows to calculate the worst-case delays and delay variations more easily. Most importantly, ATM is connection-oriented, which creates the opportunity for end systems to request a QoS level, and for the network to ensure that resources are available to deliver and maintain those QoS contracts [13].

Bandwidth	• Peak Cell Rate (PCR) • Sustained Cell Rate (SCR) • Minimum Cell Rate (MCR)
Delay and Jitter	• Cell Transfer Delay (CTD) • Cell Delay Variation (CDV)
Reliability	• Cell Loss Ratio (CLR)
Cost	• Link Weighting

Figure 1.1 Defined QoS parameters in ATM

Generally, the QoS parameters that are expected the network to supply are bandwidth, delay, delay jitter, reliability, and cost etc. In ATM, these QoS parameters can be translated into very specific ATM metrics, shown in Figure 1.1. Unlike IP network, ATM is the backbone that QoS routing can really take place.

1.2 Objective of QoS Routing

The purpose of routing in networks is to decide which output link that the incoming packets should be transmitted on. The term QoS routing specifies a routing process that guarantees quality-of-service, i.e., finding a feasible path (or tree) that meets specific QoS requirements. In the meantime, the network resources must be utilized efficiently, which is the universal goal for all kinds of networks. A network that has the QoS provisioning may benefit from many aspects. Successful QoS routing leads to a network with more efficient bandwidth allocation and less congestion.

The routing algorithm is part of the layer-three software responsible for the route selection. There are some properties that are desired for routing algorithms: correctness, simplicity, robustness, stability, fairness, and optimality [12]. Hence, these are also the guidelines for designing QoS routing algorithms.

Routing algorithms can be grouped into two major classes: non-adaptive and adaptive. Non-adaptive algorithms do not base their routing decision on measurements or estimates of the current traffic and topology, whereas adaptive ones do. If an adaptive algorithm manages to adapt well to the traffic, it will naturally outperform an algorithm that is oblivious to the network parameters. However, this is a challenging goal to achieve.

The QoS routing algorithm surely belongs to the adaptive category, which is difficult to realize. The difficulties are also from other aspects, such as network information exchange and update, the time and space complexity, coexistence of QoS routing and best-effort routing, and imprecise information [1] etc.

1.3 Contribution and Organization of the Thesis

This thesis focuses on the unicast multi-constrained QoS routing problems, studies existing source routing approaches, and classifies them as three categories. A new heuristic framework, which can integrate existing source routing algorithms and solve NP-complete QoS routing problems within polynomial time, is then proposed.

The rest of the thesis is organized as follows. Chapter 2 provides background information for the QoS routing problems. Chapter 3 summarizes existing source routing strategies. Chapter 4 introduces a new heuristic framework. Chapter 5 provides the simulation results for the proposed framework. Conclusions are drawn in Chapter 6.

1.4 Research Limits

QoS routing problem is a real-time application involving lots of practical issues. such as routing protocols, packet scheduling and resource reservation. However, the research works have to adopt some assumptions and simplifications. The simulation is done based on the mathematical models of the network. The network topology is modeled by a directed graph. In this thesis, examples and simulations use undirected graphs for simplicity. The network topology and link state information is assumed to be precise.

CHAPTER 2

PRELIMINARY

In this chapter, the mathematical models that are essential for solving QoS routing problems are introduced and the basic concepts of QoS routing are provided.

2.1 Mathematical Model

A real network is constructed by servers, hosts, routers, and connections. Routers take part in the routing protocols, and the routing information is exchanged between routers. Servers and hosts discover their preferred routers by protocols (e.g., ICMP) other than the routing protocol. A network can be abstracted from the router's point of view. Only the information of the routers and the connections between them are kept in the mathematical model for routing purposes.

The link metrics are quantitative values representing the link weights of the connections. The link metrics are the reflection of the basic characteristics of a network. The information they contain should be sufficient to support desired QoS. Metrics should be orthogonal to each other so that there should no redundant information among the metrics [6].

2.1.1 Graph Representation

Graph is an easy and intuitive way of representing a network topology.

Definition 1. A graph $G(V,E)$ consists of a set of vertices (routers) V and a set of edges (connections) E. Two vertices are adjacent if they are connected directly to each other. If they are not adjacent but the traffic can flow from one to the other via one or more intermediate vertices, the two vertices are connected.

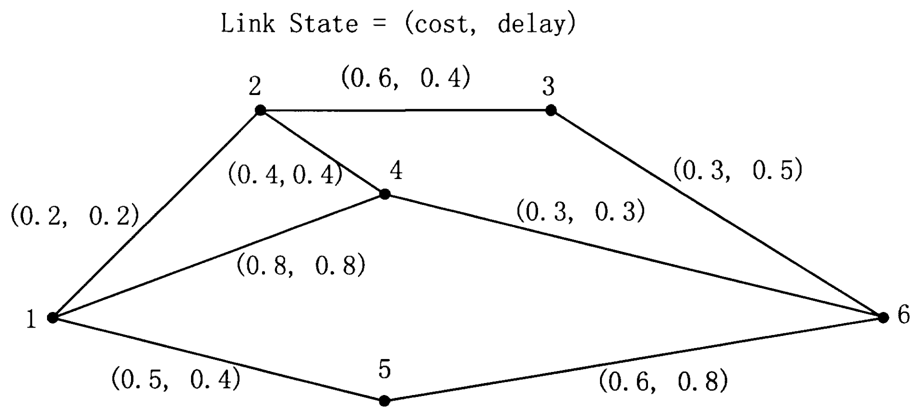

Figure 2.1 A network and link metrics modeled by a graph

A set of link metrics is associated with each edge between two adjacent vertices. Figure 2.1 presents a network graph with the link metrics of cost and delay.

2.1.2 Matrix Representation

The graph shown in Figure 2.1 can be further represented in the matrix form, namely distance matrix. A distance matrix is always square, i.e., it has the same number of rows as columns. Each vertex in the graph has one row and one column. Both the connectivity and the link metrics are reflected by the matrix elements. The network graph shown in Figure 2.1 can be represented by the distance matrix, as shown in Figure 2.2.

$$\begin{bmatrix} (0,0) & (0.2,0.2) & (\infty,\infty) & (0.8,0.8) & (0.5,0.4) & (\infty,\infty) \\ (0.2,0.2) & (0,0) & (0.2,0.4) & (0.4,0.4) & (\infty,\infty) & (\infty,\infty) \\ (\infty,\infty) & (0.2,0.4) & (0,0) & (\infty,\infty) & (\infty,\infty) & (0.2,0.5) \\ (0.8,0.8) & (0.4,0.4) & (\infty,\infty) & (0,0) & (\infty,\infty) & (0.3,0.3) \\ (0.5,0.4) & (\infty,\infty) & (\infty,\infty) & (\infty,\infty) & (0,0) & (0.6,0.8) \\ (\infty,\infty) & (\infty,\infty) & (0.2,0.5) & (0.3,0.3) & (0.6,0.8) & (0,0) \end{bmatrix}$$

Figure 2.2 Matrix representation

The matrix element M_{ij} represents a set of link metrics on the edge between the adjacent Vertex i and Vertex j. The diagonal elements are $(0,0)$ and the elements representing the edges between the non-adjacent vertices are (∞,∞).

2.1.3 QoS Constraint

The QoS requirements of a connection setup are given as a set of the end-to-end constraints. The constraints specify the demands of the network resources requested by a connection in order to ensure QoS. They can be path constraints or tree constraints [5] according to the characteristics of the connection requests. A path constraint specifies the end-to-end QoS requirements on a single path, whereas a tree constraint specifies the QoS requirements for the entire multicast tree. For instance, a delay constraint as a path constraint sets the upper bound of the delay along a path, while a delay constraint as a tree constraint requires the largest end-to-end delay from the source to any destination in the tree not to exceed an upper bound [1].

Whether a constraint is achieved or not can be computed based on the link metrics. The metrics (or constraints) are classified as three types [6].

Definition 2. Let $m(i,j)$ be a metric for link (i,j). For any path $p=(i,j,k,\cdots,p,q)$, metric m is additive if

$$m(p)=m(i,j)+m(j,k)+\cdots+m(p,q).$$

Metric m is multiplicative if

$$m(p)=m(i,j)\times m(j,k)\times\cdots\times m(p,q).$$

Metric m is concave if

$$m(p)=\min[m(i,j),m(j,k),\cdots,m(p,q)].$$

Bandwidth, cost, delay, delay jitter, and loss ratio are commonly used as the QoS constraints. Apparently, cost, delay, delay jitter are additive. Bandwidth is concave. Loss ratio can be expressed as

$$m(p) = 1 - ((1 - m(i,j)) \times (1 - m(j,k)) \times \cdots \times (1 - m(p,q))),$$

which is complicated. However, $1 - m(p)$, namely success ratio, are of the multiplicative type.

Since the multiplicative type can be converted into the additive type by the logarithmic operation and the concave type can be easily pruned by selecting the bottleneck link [6], the constraints considered in this thesis are additive, unless otherwise mentioned.

2.2 QoS Routing Issues

QoS routing has two major tasks. One is the link state information update and the other is path selection. This thesis focuses on the latter one.

2.2.1 Link State Update

Each vertex in a network maintains a local state, which includes the queuing and propagation delay, the residual bandwidth, and the availability of other resources etc. of its own [1].

The combination of the local states of all vertices is called a global state. The global state is maintained by either the link state protocol [7] or the distance-vector protocol [8], which allows information to be exchanged between the vertices in the network. The link state protocol broadcasts the local state of each vertex to every other vertex in the network, so that each vertex knows the topology of the network and the link metrics on each edge. The distance-vector protocol periodically exchanges distance vectors among the adjacent vertices. A distance vector has an entry for every possible destination, consisting of the property of the best path and the next vertex on the best path.

The performance of any routing algorithms directly depends on how well the state information is maintained. The global state kept by a vertex suffers from the delay of propagating the local states. As the network size grows, the imprecision increases. Usually, the change of the network topology is relatively infrequent comparing to those link state metrics. In this thesis, it is assumed that the state information of the network is well maintained and updated.

2.2.2 QoS Routing

Provided with the topology and the link state information of a network, it should be sufficient to decide which path that the packets should follow in order to achieve specified QoS requirements. The routing problems can be divided into two major classes: unicast routing and multicast routing. The unicast routing problem is to find a path from a single source to a single destination that satisfies the QoS constraints. The multicast routing problem is to find the best feasible tree from a single source to a set of destinations that satisfies given constraints.

2.3 Routing Schemes

From the viewpoint of goal, unicast routing can be classified as flooding, random routing and policy routing [4].

2.3.1 Flooding, Random Routing and Policy Routing

Flooding is the simplest routing technique and needs no service requirement and network information at all. A path finding packet is sent by a source node to all of its neighbors. At each intermediate node, an incoming packet is retransmitted on all outgoing links except the link from which it arrived [9]. Under flooding scheme, a termination rule is needed to stop the packet circulation. Clearly, flooding technique is highly robust and could be used to send high-priority messages or exchange link state information. However, in addition to the unbounded circulation, which increases the total traffic load and delay in the network, no QoS requirement can be guaranteed in flooding.

Random routing has the simplicity and robustness of flooding with far less traffic load. A node selects only one outgoing path at random for retransmission of an incoming packet. So the utilization of outgoing links is in a round-robin fashion. A refinement of this technique is that the selection among each link is based on the probability that is assigned to each outgoing link in advance [10]. QoS guarantee is not supported in random routing scheme either.

Policy routing incorporates policy-related constraints into path computation along with topology and link metrics of the network. The comprehensive policy can be the routing objective, optimization criterion and other strategies. Most importantly, service requirements can be properly mapped into those policy-related routing constraints, which ensures the feasibility of QoS-based routing.

2.3.2 Source Routing and Distributed Routing

In policy routing scheme, the routing strategies can be classified as the source routing and the distributed routing from the viewpoint how the state information is maintained and how the search of feasible paths is carried out.

In the source routing, each vertex maintains the complete global state, including the network topology and the link state information of every edge. Based on the global state, a feasible path is locally computed at the source vertex. A control message is then sent out along the selected path to request resource reservation if a feasible path is found. Otherwise, the connection is rejected.

In distributed routing, some algorithms also depend on the global state information maintained by each vertex more or less [6, 14], and some do not [15, 16]. The path computing decision is, however, made on a hop-by-hop basis. To avoid loops in routing is a major issue of designing the distributed algorithms.

2.4 Problem Formulation

This thesis focuses on the unicast source routing problems. The multi-constrained path (MCP) problem in this class is known to be NP-complete [2]. The MCP problem is formulated as follows:

Definition 3. Given a network graph $G(V,E)$, where V is the set of vertices and E is the set of edges, each link (u,v) is associated with a set of link weight $w_1(u,v), w_2(u,v), \cdots, w_n(u,v)$. The n constrained path problem is to find a path p from a source vertex s to a destination vertex t, such that $w_1(p) \leq C_1$, $w_2(p) \leq C_2, \cdots, w_n(p) \leq C_n$.

In order to tackle this problem, it is necessary to modify the original MCP problem with relaxed conditions or parameters. Then the path may be computed by the extended shortest path algorithms within polynomial time. The source routing heuristic algorithms will be introduced in Chapter 3, and a new source routing framework is proposed in Chapter 4.

CHAPTER 3

SOURCE ROUTING STRATEGIES

Multi-constrained QoS routing problem is challenging due to its NP-completeness. There are many source routing heuristic algorithms, but they are only approximate approaches and may not retrieve all the solutions to the original problem. However, these heuristics are necessary to make NP-complete problem tractable. This chapter introduces and summarizes existing source routing approaches for the MCP problem.

3.1 Extended Bellman-Ford Algorithm

Before introducing the heuristic algorithms that solve the MCP problem within polynomial-time, this section provides the Extended Bellman-Ford Algorithm ($EBFA$) that is capable to solve the problem regardless of the time and space complexity [11].

Definition 4. Given a directed graph $G(V,E)$ with $k \geq 2$ weight functions $w_l : 1 \leq l \leq k$, a source vertex s and a destination vertex t, a path $p = s \rightarrow v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow t$ is an optimal path from s to t if there does not exist another path q from s to t such that $w_l(q) < w_l(p)$ for any $1 \leq l \leq k$.

Figure 3.1 illustrates the algorithm. Procedure $BF(G,s,t,w,c)$ is the standard Bellman-Ford algorithm for finding the shortest path p from Vertex s to Vertex t, such that $w(p) < c$. However, Procedure $RELAX(u,v,w)$ is modified for multi-constraint situation. Line (16) – (20) initializes the variables and line (21) – (25) performs the relax operation.

```
(1)  $RELAX(u,v,w)$
(2)  for each $w(p)$ in $PATH(u)$
(3)    flag = 1
(4)    for each $w(q)$ in $PATH(v)$
(5)      if$(w_l(p)+w_l(u,v) \geq w_l(q))$ for all $1 \leq l \leq k$ then
(6)        flag = 0
(7)      end if
(8)      if$(w_l(p)+w_l(u,v) < w_l(q))$ for any $1 \leq l \leq k$ then
(9)        remove $w(q)$ from $PATH(v)$
(10)     end if
(11)   end for
(12)   if(flag = 1) then
(13)     add $w(p)+w(u,v)$ to $PATH(v)$
(14)   end if
(15) end for

(16) $BF(G,s,t,w,c)$
(17) for $i=0$ to $|V|-1$
(18)   $PATH(i)=\varnothing$
(19) end for
(20) $PATH(s)=\{0\}$
(21) for $i=1$ to $|V|-1$
(22)   for each edge $(u,v) \in E$
(23)     $RELAX(u,v,w)$
(24)   end for
(25) end for
(26) for each $w(p)$ in $PATH(t)$
(27)   if$(w(p)<c)$ then
(28)     return "success"
(29)   end if
(30) end for
(31) return "fail"
```

Figure 3.1 Extended Bellman-Ford Algorithm

Finally, line (26) – (31) checks if the paths obtained from the relax operation are feasible paths to the problem. Procedure $RELAX(u,v,w)$ checks if the there is a better optimal path as hop count increases. Path p is maintained by current Vertex u, and path

q is maintained by Vertex v, which is one hop away from u. If $w_l(p)+w_l(u,v) \geq w_l(q)$ for all $1 \leq l \leq k$, path q is kept in $PATH(v)$ at Vertex v. If $w_l(p)+w_l(u,v) < w_l(q)$ for any $1 \leq l \leq k$, a new optimal path $P+u \rightarrow v$ will be insert to $PATH(v)$ at Vertex v. By iteratively doing so, all paths will be checked as hop count increases.

The problem behind this approach is the extremely large time complexity. The standard Bellman-Ford algorithm has running time of $O(|V||E|)$. Procedure $RELAX(u,v,w)$ takes consideration of each optimal path in $PATH(u)$ and $PATH(v)$ for every constraint w_l, $1 \leq l \leq k$. The number of these optimal paths maintained at each vertex can be exponential. Hence, the running time of *EBFA* may grow exponentially although it is guaranteed to find the feasible path if there exists one.

In order to reduce the time complexity, some constraints or conditions of the original problem can be modified. In the following sections, three heuristic approaches will be introduced with paradigms. They may or may not find a feasible path even if there exists one. The performance usually relies on the parameter that has been compromised.

3.2 Limited Path Heuristic

The exponential number of optimal paths maintained at each vertex leads to the NP-completeness of a multi-constrained QoS routing problem. Intuitively, if the optimal paths maintained at each vertex are set to a limited number [11], the overall time complexity will be reduced to polynomial time.

Procedure $RELAX(u,v,w)$ in Figure 3.1 is then be modified as shown in Figure 3.2. A new optimal path will be inserted to $PATH(v)$ only when the size of $PATH(v)$ is not greater than a user-defined integer X. By doing so, the overall running time of the *EBFA* algorithm is reduced to $O(X^2|V||E|)$.

```
$RELAX(u,v,w)$
for each $w(p)$ in $PATH(u)$
  flag = 1
  for each $w(q)$ in $PATH(v)$
    if($w_l(p)+w_l(u,v) \geq w_l(q)$) for all $1 \leq l \leq k$ then
      flag = 0
    end if
    if($w_l(p)+w_l(u,v) < w_l(q)$) for any $1 \leq l \leq k$ then
      remove $w(q)$ from $PATH(v)$
      $|PATH(v)| = |PATH(v)| - 1$
    end if
  end for
  if(flag = 1 $$ $|PATH(v)| \leq X$ ) then
    add $w(p)+w(u,v)$ to $PATH(v)$
  end if
end for
```

Figure 3.2 Procedure RELAX(u, v, w) for the limited path heuristic

The probability of finding a feasible path when there exists one is related to the value of X. The larger X is, the better the performance of path finding is, at a cost of the time complexity of course.

3.3 Limited Granularity Heuristic

Suppose there are two additive constraints, e.g. delay and cost. Both link metrics are positive real numbers. *EBFA* algorithm tries to search the solutions by enumerating the solution space for both constraints. If one of the constraints can be mapped or discretized to positive integers, the search for paths is limited to one constraint while performing finite looping on the other one. As known, single constrained path problem has polynomial time solution. Hence, the modified problem can be solved within polynomial time.

```
$RELAX(u,v,w)$
for each $d[v,i]$ in $PATH(u)$
  Let $j$ be the largest for $C_j < C_i - w_2(u,v)$
  if $(d[v,i] > d[u,i] + w_2(u,v))$ then
    $d[v,i] = d[u,i] + w_2(u,v)$
  end if
end for

$EBF(G,s,t,w,c)$
for $i=0$ to $|V|-1$
  for $j=1$ to $X$
    $d[i,j]=\infty$
  end for
end for
for $j=1$ to $X$
  $d[s,j]=0$
end for
for $i=1$ to $|V|-1$
  for each edge $(u,v) \in E$
    $RELAX(u,v,w)$
  end for
end for
if $(d[s,X] < C_2)$ then
  return "success"
end if
return "fail"
```

Figure 3.3 Pseudo code for the limited granularity heuristic

S. Chen and K. Nahrstedt proposed a granularity heuristic algorithm [18] for $MCP(G,s,t,w_1,w_2,C_1,C_2)$. First, one constraint, say $w_2 \in R_0^+$, is mapped to $w_2' \in I^+$ by function:

$$w_2'(u,v) = \left\lceil \frac{w_2(u,v)X}{C_2} \right\rceil$$

where X is a tunable positive integer.

In other words, the limited granularity heuristic rounds up the range $[0, c_2]$ into a finite range with X elements, $C_1, C_2, \cdots, C_X$, evenly, where $0 < C_1 < C_2 < \cdots < C_X = C_2$. Therefore, the link state metric can only take one of X discrete values. The round up operation ensures that a solution to the modified problem is also a solution to the original problem. The modified problem can be solved by the Extended Bellman-Ford algorithm within polynomial time, as shown in Figure 3.3.

The running time of procedure $RELAX(u,v,w)$ is $O(X)$. Thus, the time complexity of this algorithm is $O(X|V||E|)$. If $w_2(p) \leq [1-(|V|-1/X)]C_2$, the limited granularity heuristic guarantees to return a feasible path. For example, if choose $X = 10|V|$, the path with the weight $w_2(p) \leq 0.9C_2$ can be found.

3.4 Constraint Aggregation

Having introduced two heuristic approaches that depend on a tunable parameter X to reduce the time and space complexity, this section reviews the original problem from a different point of view. The NP-completeness of the original problem comes from the multiple constraints. If these constraints can be aggregated to one constraint by some way, the problem is solvable within polynomial-time.

This is so called constraint aggregation approach. This approach originates from Jaffe's work [2] in 1984. He proposed an alternative way to tackle the two-constrained QoS routing problem. Two cost functions are replaced by one cost function that combines them linearly. The new cost function is:

$$w(p) = \alpha w_1(p) + \beta w_2(p)$$

where α and β are positive integers.

This relationship can be described as a set of equal-length lines in a plane, as shown in Figure 3.4. The black dots represent the costs of all the paths between the source s and the destination t in two-dimension. The shaded region represents where the solutions should fall. The equal-length lines represent different constant values of $\alpha w_1(p) + \beta w_2(p)$. Each of the parallel lines intersects solutions with equal length $const$.

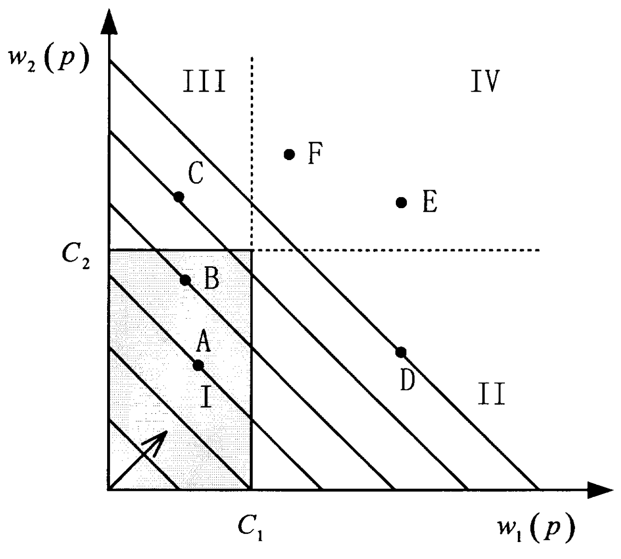

Figure 3.4 Demonstration of Solution Regions

All solutions lying above a certain line have length larger than those ones below or on the line [19]. Dots within Region I, i.e., A and B, represent the feasible paths to the original problem. Dots within Region II and III, i.e., C and D, represent the paths that violate one constraint. Dots within Region IV, i.e., E and F, represent the paths that violate both constraints.

The standard shortest path algorithm selects the path represented by the dots that the equal-length line first intersects starting from the origin, based on one aggregated cost function $w(p)$. Assume $\alpha = \beta = 1$ as shown in Figure 3.4, the path represented by A in will be selected. However, there are situations that the shortest path algorithm may not reach a feasible path. Figure 3.5 shows an example.

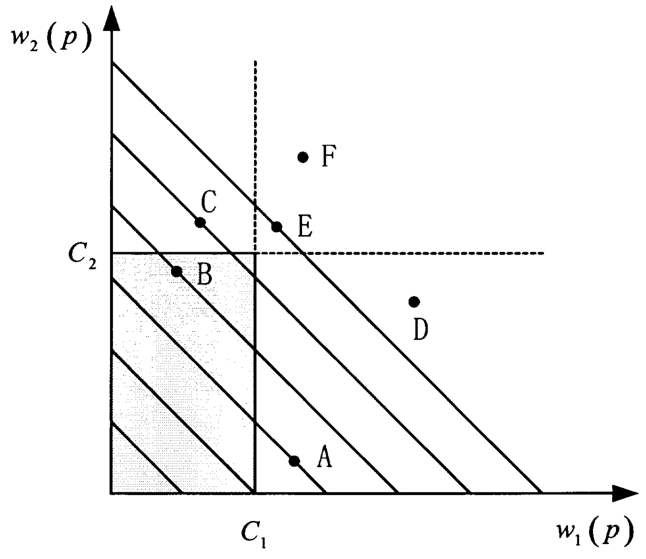

Figure 3.5 Demonstration of selecting a non feasible path

The shortest path algorithm will first intersect A based on the aggregated cost function $\alpha w_1(p) + \beta w_2(p)$ if α and β have the same values as the example shown in Figure 3.4. But A does not represent a feasible path since its w_1 weight violates constraint C_1.

Figure 3.6 demonstrates that a feasible path can be found by setting $\alpha = 2$ and $\beta = 1$ for the same situation in Figure 3.5.

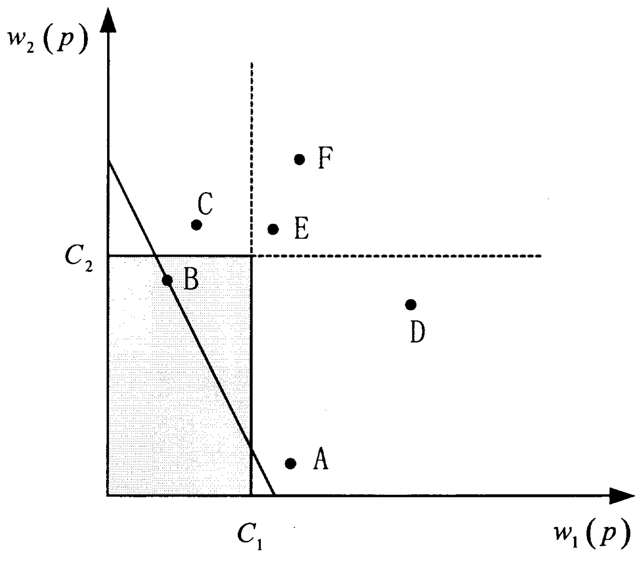

Figure 3.6 Demonstration of retrieving a feasible path

This implies that the slope of the equal-length lines is critical for this heuristic approach. Many works [4, 20] have been proposed on how to adjust this slop for the purpose of finding a feasible path with higher probability.

CHAPTER 4

A NEW HEURISTIC SOURCE ROUTING FRAMEWORK

Having observed existing source routing approaches to solve the MCP problem, a new source routing framework is proposed in this chapter.

4.1 Motivation

Most of the heuristic algorithms for solving the MCP problem try to find a path subject to relaxed constraints or conditions in order to reduce the complexity. Thus, some feasible paths may become irretrievable.

For example, given a network graph and the link metrics as shown in Figure 4.1, find a path that satisfies the given end-to-end constraints $(c,d)=(1,1)$ from Vertex 1 to Vertex 6.

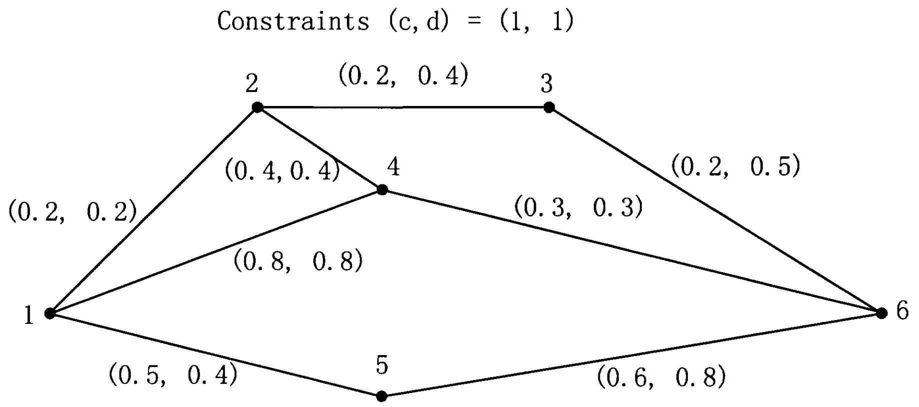

Figure 4.1 Network graph with constraints

Method 1: Using the aforementioned constraint aggregation approach, a new cost function $w=c+d$ is defined. The shortest path algorithm will select the path $1 \rightarrow 2 \rightarrow 3 \rightarrow 6$ with an aggregated weight of 1.7. Since the actual weights of the path are

$(0.6, 1.1)$, which violates the original end-to-end constraints, it is not a feasible path for the problem.

Method 2: Before applying the path selection algorithm, expand the destination from Vertex 6 to its neighboring vertices, i.e. Vertex 3, 4, and 5. Then the constraints are revised for Vertex 3, 4, and 5 by subtracting the link metrics $(0.2, 0.5)$, $(0.3, 0.3)$ and $(0.6, 0.8)$ from the original constraints $(1,1)$, accordingly. The network graph is updated as shown in Figure 4.2. The dotted lines represent edges that have been eliminated and the hollow vertex represents the removed vertex.

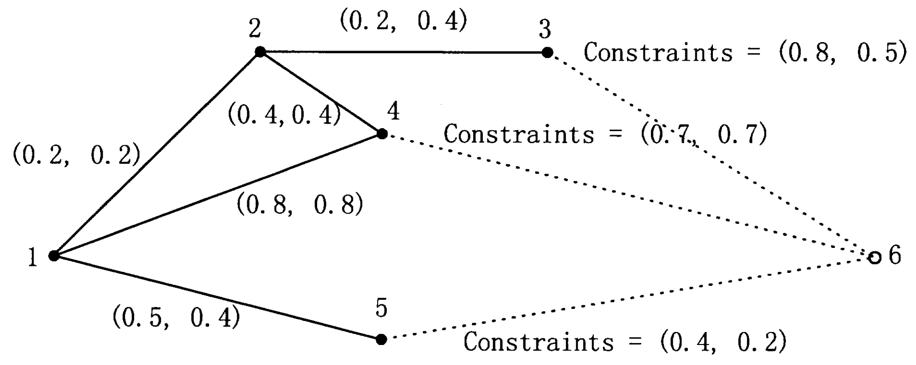
2
(0. 2, 0. 4)
3
Constraints = (0. 8, 0. 5)
(0. 4, 0. 4)
4
Constraints = (0. 7, 0. 7)
(0. 2, 0. 2)
(0. 8, 0. 8)
1
6
(0. 5, 0. 4)
5
Constraints = (0. 4, 0. 2)

Figure 4.2 Network graph with 1-hop expansion

Our modified problem is now to find a feasible path that satisfies the revised constraints from Vertex 1 to any of Vertex 3, 4, or 5. Note that the feasible paths from Vertex 1 to Vertex 3, 4, or, 5 are also the feasible paths of the original problem. By applying the same strategy in Method 1, the path $1 \to 2 \to 4$ is selected. Vertex 4 is known to have been expanded from Vertex 6, thus $1 \to 2 \to 4 \to 6$ is the completed path from Vertex 1 to Vertex 6, which turns out to be a feasible path.

Continuing Method 2, now expand two hops from Vertex 6 and revise the constraints. Both Vertex 3 and 4 will be expanded to Vertex 2. The revised constraints are

$(0.6, 0.1)$ from Vertex 3 to Vertex 2, and $(0.3, 0.3)$ from Vertex 4 to Vertex 2. Both of them need to be kept at Vertex 2, as shown in Figure 4.3. If both constraints in one set are tighter than those in the other set, $(0.2, 0.2)$ and $(0.3, 0.3)$ for instance, the tighter one, $(0.2, 0.2)$, can be pruned.

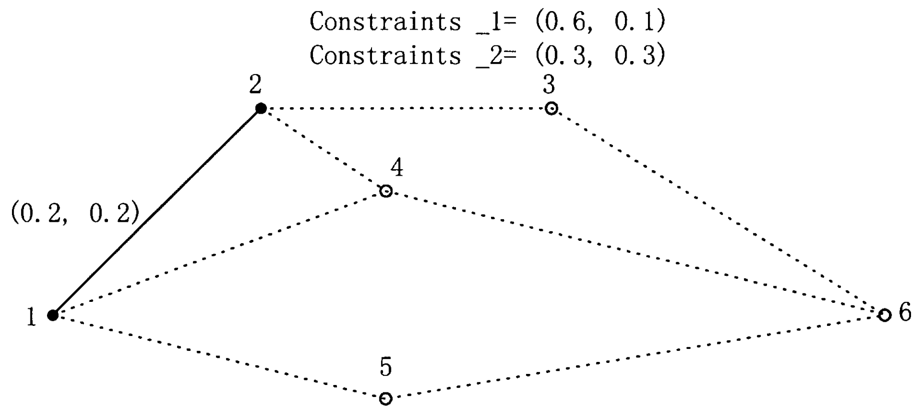

Figure 4.3 Network graph with 2-hop expansion

As compared with Method 1, it is observed that the modified problem has multiple destinations instead of one destination in the original problem. Hence, the probability of finding a feasible path is increased. Although the expansion of the destinations introduces a computational overhead to the network, in return, it keeps more accurate link state information. As a matter of fact, the complexity of the single source multi-destination algorithm is not necessarily higher than that of the single source single destination algorithm [3]. If the complexity introduced by the expansion can be tolerated, it is possible to have a better chance to find a feasible path at the source routing stage. It also enhances scalability of the source routing to some extent.

4.2 Complexity Analysis

Usually, the computational complexity of source routing relies on either the number of vertices or edges, or both of them, e.g., $O(|V||E|)$ for the Bellman-Ford algorithm. It can be observed that the expansion operation in Method 2 will eliminate some edges and vertices but possibly introduce more sets of constraints on the same vertex. Hence, the computational complexity of Method 2 is the reduced complexity of the source routing due to the reduced number of vertices and edges, plus the complexity introduced by the expansion operation and the complexity introduced by multiple sets of constraints at some vertices.

In order to achieve similar complexity as Method 1, the number of hops allowed to expand from the destination is critical in Method 2.

Since a randomized network is difficult to analyze, an $N \times N$ mesh network is used to illustrate the impact of the expansion operation on the performance of the overall source routing. Expand the destination vertex to its neighboring vertices that are n hops away and observe how the number of network vertices and edges changes according to Method 2. Assume the network is large enough so that the edge effect is negligible. Figure 4.4 shows a vertex (center) has been expanded 3 hops. Dotted lines represent eliminated edges and hollow nodes represent removed vertices. Arrows illustrate the direction of propagation. Solid nodes reveal the boundary of the expansion operation.

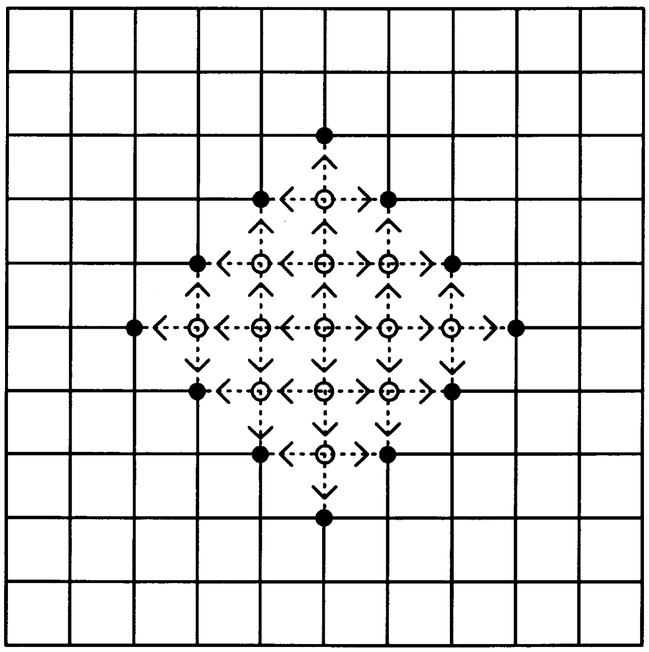

Figure 4.4 3-hop expansion in a mesh network

Lemma 1. For a large $N \times N$ mesh network, the worst-case scenario of expanding n hops from the destination vertex introduces $4 \times 3^{n-1}$ extra sets of constraints and eliminates $2n^2 - 2n + 1$ vertices.

Proof: The number of eliminated vertices is:

$$1 + 4\sum_{i=2}^{n}(i-1) = 1 + 4(\frac{n(n-1)}{2}) = 2n^2 - 2n + 1$$

Each vertex will be expanded to three directions except that the origin is expanded to four directions. Hence, the number of introduced sets of constraints is:

$$4 \times 3^{n-1}$$

QED.

Lemma 2. For a large $N \times N$ mesh network, the edges that are eliminated by expanding n hops from the destination vertex are $4n^2$.

Proof: The number of the eliminated edges represented by dotted lines in Figure 4.4 can be counted as:

$$2 \times \sum_{i=1}^{2n} i - 2n = 4n^2$$

QED.

Lemma 3. For a large $N \times N$ mesh network, the computational complexity of the Bellman-Ford algorithm is $O(2N^3(N-2))$.

Proof: The Bellman-Ford algorithm has computational complexity of $O(|V||E|)$, where $|V|$ is the number of vertices and $|E|$ is the number of edges. In an $N \times N$ mesh network, $|V| = N^2$ and $|E| = 2N(N-2)$. Hence, $O(|V||E|) = O(2N^3(N-2))$.

QED.

Theorem 1. The time complexity of n-hop-expansion based Extended Bellman-Ford algorithm is $O((N^2 - 2n^2 + 2n - 1)(2N^2 - 4N - 4n^2) + 16 \times 3^{n-1})$.

Proof: The overall complexity after n-hop expansion is:

$$\begin{aligned}
&O((|V| - |V|_{removed})(|E| - |E|_{removed})) + O(checking) \\
&= O((N^2 - (2n^2 - 2n + 1))(2N(N-2) - 4n^2) + 4 \times 4 \times 3^{n-1}) \\
&= O((N^2 - 2n^2 + 2n - 1)(2N^2 - 4N - 4n^2) + 16 \times 3^{n-1})
\end{aligned}$$

QED.

Figure 4.5 plots the difference in the algorithm running time between Method 2 and Method 1 versus the number of hops to expand in a 7×7 mesh network.

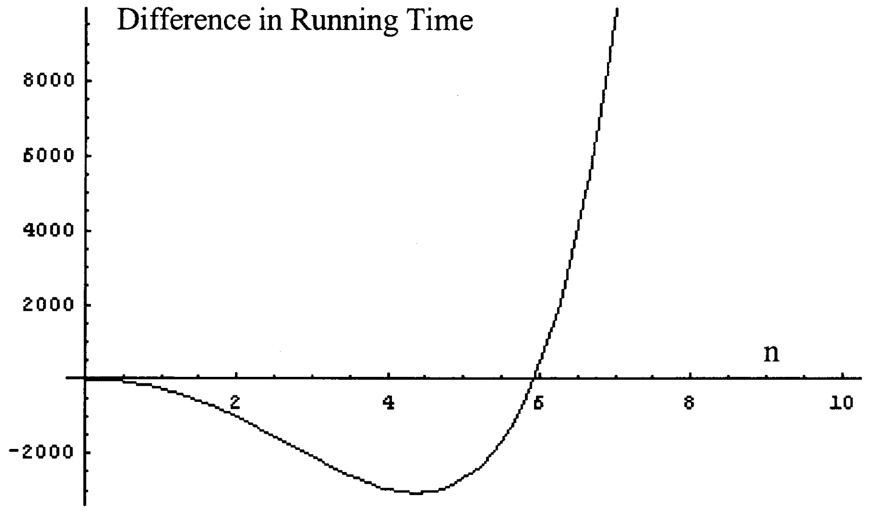

Figure 4.5 Difference in running time between two methods

The zero-crossing point is around $n = 6$, implying that Method 2 has less computational complexity than Method 1 as long as $n < 6$. So, Method 2 is able to achieve similar complexity as Method 1 if n is set properly.

4.3 Mesh Network Versus Randomized Network

The impact of Method 2 on the computational complexity has been observed in a mesh network. Now apply this method to randomized network. A question that arises is how similar the randomized network is to the mesh network having been observed. The average hop count of the shortest paths from a random source and random destination can be computed in a mesh network, and it can be simulated for a randomized network.

Theorem 2. The average hop count of the shortest paths from a random source and random destination in an $N \times N$ mesh network is approximate $\frac{2}{3}(N-1)$.

Proof: Choose a source vertex (x_1, y_1) and a destination vertex (x_2, y_2) randomly from an $N \times N$ mesh network, of which the coordinates of the most southwest vertex is $(1,1)$ and the most northeast vertex is (N,N) respectively. The number of the average hop counts between these two vertices is:

$$E[|x_2 - x_1| + |y_2 - y_1|] = 2E[|x_2 - x_1|] = \frac{2}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}|i-j| = \frac{4}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{i}|i-j|$$

Assume $i - j = k$

$$= \frac{4}{N^2}\sum_{i=1}^{N}\sum_{k=0}^{i-1}k = \frac{4}{N^2}\sum_{i=1}^{N}\frac{i(i-1)}{2} = \frac{2}{N^2}\left[\frac{N(N+1)(2N+1)}{3} - N(N+1)\right]$$

$\approx \frac{2}{3}(N-1)$, when N is large.

QED.

It is difficult to derive this parameter in a randomized network. Instead, the average hop count (hc) of the shortest paths from a random source and random destination is simulated in randomized networks of different sizes and compare the results with mesh networks. The simulation results show that:

- For 50-node randomized network, $hc = 3.7$ whereas for a 7×7 mesh network, $hc = 4$ according to Theorem 2.
- For a 100-node randomized network, $hc = 5.6$ whereas for a 10×10 mesh network, $hc = 6$ according to Theorem 2.

The comparison shows that both mesh network and randomized network have similar property in terms of average hop count of the shortest path. Randomized network actually has slightly better performance for this case. Therefore, it should be confident to apply Method 2 to a randomized network with no additional complexity by setting n properly, even though the complexity is analyzed in a mesh network.

4.4 Purposed Source Routing Framework

The MCP problem contains multiple additive constraints. For illustrative purposes, consider cost and delay constraints, and formulate the cost-delay-constrained problem as follows:

Definition 5. Given a directed graph $G(V,E)$, a source vertex s, a destination vertex t, a cost function c, a delay function d, a cost bound C and a delay bound D, the $MCP(G,s,t,c,d,C,D)$ problem is to find a path p, from s to t such that the total cost $c(p) \leq C$ and the total delay $d(p) \leq D$.

The proposed source routing algorithm, Source Routing Destination Expansion ($SRDE$), can be described as a shortest path routing algorithm based on n-hop expansion fro the destination. The routing procedure is performed in three steps.

Step 1. The algorithm expands n hops from the destination t. It records all vertices t', which are n hops away from t, and paths p', which are sets of edges from t to t'. C' and D' are the cost and delay of p', i.e. $C' = c(p')$ and $D' = d(p')$. The edges and vertices along p' are eliminated from the network graph.

Step 2. The algorithm performs the shortest path selection to solve the modified problem $MCP(G,s,t',c,d,C-C',D-D')$, which is to find a feasible path p'' from a single source s to any of the multiple destinations t' with revised cost and delay constraints $C-C$ and $D-D'$.

Step 3. Combine p' and p'' to form a path p.

Theorem 3. Path $p = p' \cup p''$ is a solution to the original $MCP(G,s,t,c,d,C,D)$ problem.

Proof: $c(p'') \leq C - C'$ and $d(p'') \leq D - D'$ since p'' is a solution to $MCP(G, s, t', c, d, C - C', D - D')$. Also $C' = c(p')$ and $D' = d(p')$ from step 2. Thus:

$$c(p) = c(p') + c(p'') \leq C - C' + C' = C$$
$$d(p) = d(p') + d(p'') \leq D - D' + D' = D$$

Hence, path p satisfies both constraints of the original problem.

QED.

Theorem 3 validates the algorithm *SRDE*. The Pseudo code of the algorithm is given in Figure 4.6.

Procedure $Expand(G, t, n)$ performs a "flood and prune" operation. It computes paths, which contain n hops starting from t. It prunes a path if both of the constraints are larger than the previously recorded values. Otherwise, it inserts the path as a new entry. It removes vertices and edges along those paths from the network graph G. It also returns a set of new destination vertices Dst and their corresponding constraints $con(p)$.

Procedure $SRDE(G, s, t, constraints)$ can employ any ***shortest path algorithm*** $(G, s, Dst, con(p))$ (line 27) that solves the MCP problem with a given network G, a source vertex s, a destination Dst, and a set of constraints $con(p)$. It returns "success" if a feasible path is selected.

```
(1)  $Initialize(G, t, constraints)$
(2)  for each vertex in $G$
(3)    $PATH(v) = \varnothing$
(4)  end for
(5)  $PATH(t) = \{t\}$
(6)  set the constraint of path $p = \{t\}$ to $(C, D)$
(7)  $Dst = \{t\}$

(8)  $Expand(G, t, n)$
(9)  for $i$ from 1 to $n$
(10)   for each vertex in $Dst$
(11)     for each path $p \in PATH(v)$ and its
         corresponding constraints $con(p)$
(12)       if $(con(p) - (c(u,v), d(u,v)) > (0,0))$
(13)         $Dst = Dst \cup \{u\}$
(14)         $p = p \cup \{u\}$
(15)         $PATH(v) = PATH(v) \cup p$
(16)         $con(p) = con(p) - (c(u,v), d(u,v))$
(17)       end if
(18)       remove edge $(u,v)$ from $G$
(19)     end for
(20)   end for
(21) end for
(22) remove dissociating vertices from $G$
(23) return $PATH(v)$ , $Dst$ , $con(p)$

(24) $SRDE(G, s, t, constraints)$
(25) $Initialize(G, t, constraints)$
(26) $Expand(G, t, n)$
(27) shortest path algorithm$(G, s, Dst, con(p))$
(28) for each vertex $v \in Dst$
(29)   for each path $p \in PATH(v)$
(30)     if the weights of the least cost path
         from $s$ to $v$ less than $con(p)$
(31)       return "success"
(32)     end if
(33)   end for
(34) end for
(35) return "fail"
```

Figure 4.6 Pseudo code for the SRDE algorithm

CHAPTER 5

SIMULATION RESULTS

This chapter provides the simulation results of our algorithm in the mesh network and randomized network of different sizes. Shortest path search is performed by the Bellman-Ford algorithm according to an aggregated linear cost function $w = c + d$. The results are compared with the standard Bellman-Ford algorithm. In the simulation, two QoS constraints are set to be equal, of which the range is from 0.5 to 5.5 with an increment of 0.2.

The success ratio (SR) is defined as:

$$SR = \frac{\textit{Total number of success request of the algorithm}}{\textit{Total number of success request of the optimal algorithm}}$$

Figure 5.1 shows the simulation results of 2-hop $SRDE$ in a 7×7 mesh network with comparison to the standard Bellman-Ford algorithm. The result shows that 2-hop $SRDE$ outperforms the standard Bellman-Ford Algorithm.

It has been already shown in Chapter 4 that in a 7×7 (49 vertices) mesh network, when the number of hops allowed to expand $n < 6$, the computational complexity will not increase. Based on the conservative estimation, the simulation is run for 2-hop and 3-hop $SRDE$ respectively for a 50-node randomized network topology. The results are shown in Figure 5.2.

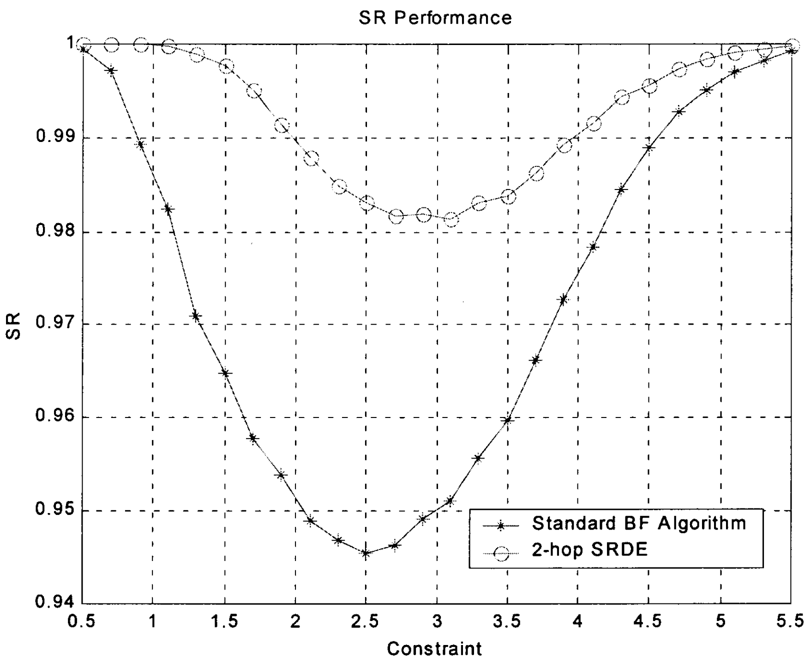

Figure 5.1 Simulation results for a 7x7 mesh network

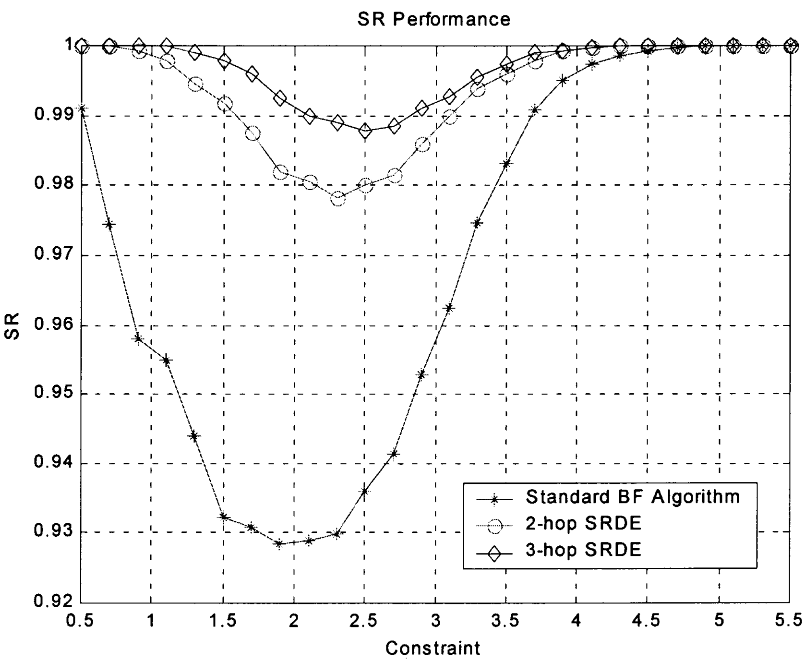

Figure 5.2 Simulation results for a 50-node network

The plot shows that the *SRDE* algorithm achieves higher success ratio for both cases of $n = 2$ and $n = 3$. Note the dramatic improvement even with $n = 2$. It implies the larger the number n is, the higher the success ratio the algorithm may achieve.

CHAPTER 6

CONCLUSIONS

This thesis has introduced a source routing framework that can effectively solve the MCP QoS routing problem. The *SRDE* algorithm expands n hops from the destination before performing the path selection algorithm. By applying this method, the original single source single destination problem becomes a single source multi-destination problem. A feasible path is searched from the source to any of the destinations. Hence, a higher success ratio is expected. Existing source routing algorithms can be integrated into this framework, resulting in better performance with no additional complexity if n is chosen properly.

REFERENCES

[1] S. Chen and K. Nahsted, "An overview of quality of service routing for next-generation high-speed network: problems and solutions," *IEEE Network*, vol. 12, no. 6, pp. 64-79, December, 1998.

[2] J. M. Jaffe, "Algorithms for Finding Paths with Multiple Constraints," *Networks*, vol 14, pp. 95-116, 1984.

[3] T. H. Cormen, C. E. Leiserson, and R. L. Rivet. *Introduction to Algorithms, Massachusetts: MIT Press*, pp. 514-56, 2000.

[4] A. Juttner, B. Szyiatovszki, I. Mecs, and Rajko, "Lagrange relaxation based method for the QoS routing problem," *Proc. IEEE INFOCOM 2001*, vol. 2, pp 859-868, 2001.

[5] W. C. Lee, M. G. Hluchyi, and P. A. Humblet, "Routing Subject to Quality of Service Constraints Integrated Communication Networks," *IEEE Network*, July/August 1995.

[6] Z. Wang and J. Crowcroft, "Quality-of-Service Routing for Supporting Multimedia Applications," *IEEE J. Selected Areas in Communications*, vol.14, pp. 1228-1234, Sep. 1996.

[7] M. Steenstrup, "Inter-Domain Policy Routing Protocol Specification: Version 1," *Internet Draft*, May 1992.

[8] C. Hedrick, "Routing Information Protocol," RFC 1058, Network Information Center, SRI International, Menlo Park, CA, June 1988.

[9] H. Pung, J. Song, and L. Jacob, "Fast and Efficient Flooding Based QoS Routing Algorithm," *IEEE Conference on Computer Communication and Network*, pp. 298-303, 1999.

[10] S. Nelakuditi, Z. Zhang, and R. T, "Adaptive Proportional Routing: A Localized QoS Routing Approach," *IEEE INFOCOM 2000*, vol 3, pp. 1566-1575, 2000.

[11] X. Yuan, "On the Extended Bellman-Ford Algorithm to Solve Two-Constrained Quality of Service Routing Problems," *IEEE Proc. Eighth International Conference on Computer Communications and Networks*, pp.304-310, 1999.

[12] A. S. Tanenbaum, *Computer Networks, New Jersey: Prentice-Hall*, pp. 196-214, 1981.

[13] J. Walrand and P. Varaiya, *High-Performance Communication Networks, California: Morgan Kaufmann Publishers*, pp. 197-233, 1996.

[14] H. F. Salama, D. S. Reeves, and Y. Viniotis, "A Distributed Algorithm for Delay-Constrained Unicast Routing," *IEEE INFOCOM 97*, April 1997.

[15] Q. Sun and H. Langendorfer, "A New Distributed Routing Algorithm with End-to-End Delay Guarantee. Unp

[16] S. Chen and K. Nahrstedt, "Distributed Quality-of-Service Routing in High-Speed Networks Based on Selective Probing," *Technical Report, University of Illinois at Urbana-Champaign, Department of Computer Science*, 1998.

[17] K. G. Shin and C.-C. Chou, "A Distributed Route-Selection Scheme for Establishing Real-Time Channel," *HPN 95*, pp. 319-329, Sep. 1995.

[18] S. Chen and K. Nahrstedt, "On Finding Multi-Constrained Paths," *IEEE ICC 98*, June 1998.

[19] H. D. Neve and P. V. Mieghem, "A Multiple Quality of Service Routing, Algorithm for PNNI," *IEEE Proc. ATM Workshop*, pp. 324-328, 1998.

[20] T. Korkmaz, M. Krunz, and S. Tragoudas, "An efficient algorithms for finding a path subject to two additive constraints," *ACM SIGCOMM 2000*, pp 318-327, June 2000.