Chapter 7

Source versus Heuristic QoS Routing Algorithms
Chapter 7

Source versus Heuristic QoS Routing Algorithms

"It is not the quantity but the quality of knowledge which determines the mind's dignity."

Unknown

7.1 Introduction

The Internet Community today needs multimedia communication than the traditional, well known Internet Protocol (IP) infrastructure. IP has proven to be very effective since no privileged service classes are considered, but it is not adequate for the introduction of “Quality of Service” (QoS). QoS communications on the internet are high speed connections and the goal of QoS routing is to find routes that satisfy the requirements of individual flows and to provide quality multimedia communications [4].

In Computer communication networks, data packets are forwarded between different nodes. Multiple paths may exist from the start node to the destination node. Due to the different qualities of links and traffic load variations, different paths may have different performance related Quality of Service properties, such as delay, response time, overhead, etc. Finding a path that satisfies some given QoS constraints is the base of QoS routing [104].

The routing problems can be divided into two major classes as mentioned in Chapter 1. They are, 1. unicast routing and 2. multicast routing [104]. The unicast routing problem is as follows. Given a source node sr, a destination node dn, a set of QoS constraints qc and an optimization goal (optional), find the best feasible path from sr to dn, which satisfies qc. The multicast routing problem is as follows. Given a source node sr, a
set of destination nodes $st$, a set of constraints $cts$ and an optimization goal (optional), find the best feasible path covering $sr$ and all nodes in $st$, which satisfies $cts$.

This Chapter presents two unicast QoS based algorithms called as Source Routing [48], [105] and the Heuristic Routing algorithm proposed by us. A Client Server based model has been generated to study the performance of the two algorithms with respect to the message overhead, response time and path delay. The Experiments and the results are analyzed.

7.2 Source Routing

Source routing [48], [105] is a method of transferring a packet through a network in which the path is already determined by the source. The information about the packet is stored in the packet itself in source routing. Whenever a packet arrives at a switching device, the decision to forward it to the next node is not needed since they are all stored in the packet itself. The device simply looks at the path information header in the packet in order to determine the port on which it should forward the packet. Source routing denotes that the source knows about the topology of the network, and hence can specify a path.

7.3 Heuristic Routing

Heuristic routing is a Routing method in which the data, such as path delay, response time and message overhead, extracted from incoming packets during specific time intervals and with different loads are used to determine the optimum routing for transmitting data back to the source. A heuristic routing scheme called as the Label Based Probing is proposed in this work.
In this routing, the source is not expected to have all the information about how to get from the source to the destination. It is sufficient for the source to know only about how to get to the next node, and so on until the destination is reached. The processing in the nodes in between the source and the destination in this case is more complicated. It has only the address of the destination rather than a complete specification of the route by the source node.

The working of the Label based distributed routing algorithm is as follows. When a connection request arrives at the source node, a certain number of “I” labels are generated and a probe packet with the “I” labels is sent to the destination in search of paths that satisfy the QoS constraints. Each probe packet carries one or more labels. Whenever an intermediate node receives a probe packet, it does the following. It searches the routing table and finds the next node that can establish connections for the request. Then it calculates the number of labels to be distributed for each of the next nodes, and finally sends a probe packet with the labels to each of the next nodes. A probe fails if there is no outgoing link that can satisfy the QoS requirement. When a probe packet reaches the destination, a path that can satisfy the QoS requirement is found. The algorithm controls the message overhead by manipulating the number of labels. Since intermediate nodes only distribute the labels but not generate any new labels, the maximum number of probe packets at any time is bounded by number of labels. Since each probe packet probes a path, the maximum number of paths probed is also bounded by the number of labels. Connecting a label based routing and forward reservation is simple. Whenever a connection request arrives at the source, the source node immediately sends a reservation packet with “I” labels to the destination. The reservation packet goes through the network and reserves resources along the path. A reservation packet fails when there is no available resource to
support the QoS requirement. When the reservation packet reaches the destination, the
destination sends an accept packet to inform the source and the intermediate nodes that this
connection has been established. When the accept packet reaches the source, the source
can start sending data. Once the source finishes sending data, it sends a release packet to
release the resources along the path for the connection.

7.4 The Network

A point to point communication network is represented as a directed, connected,
simple network $N = (V, E)$, where $V$ is a set of nodes and $E$ is a set of directed links. A
node $i$ is assumed to have the local state about all outgoing links. The state information of
link$(i,j)$ includes three parameters [106]. They are given as follows.

1) delay $(i,j)$ is the delay of the link, including propagation delay and the queuing delay.

2) bandwidth $(i,j)$ is the unused bandwidth of the link and

3) cost $(i,j)$ which can a hop count or a function of the link utilization.

The delay, bandwidth, and cost of a path $P = i - j \ldots k - l$ are defined as follows:

$$\text{delay}(P) = \text{delay}(i,j) + \ldots + \text{delay}(k,l)$$

$$\text{bandwidth}(P) = \{\text{bandwidth}(i,j), \ldots, \text{bandwidth}(k,l)\}$$

$$\text{cost}(P) = \text{cost}(i,j) + \ldots + \text{cost}(k,l)$$

7.5 Source QoS Routing Algorithm

The algorithm followed in the client server model for Source QoS Routing
algorithm [48][105] is as follows.

1. The client router first tries to find two paths, one that is going out of the server and
the other, a path that is coming into the server. This is found in this model using the
Dijkstra's Shortest Path Algorithm. There is a need to find two paths since the QoS requirements may be different for each direction. For example, the server might be a multimedia file that responds to requests from the client that denotes that the incoming path needs more bandwidth allocated to it than its outgoing line. The outgoing and incoming paths could be formed by different links since traffic and resource conditions are not the same usually in both directions.

2. If any one of the path (outgoing or incoming) is not found, then the client is informed and the connection process stops.

3. Otherwise, the client router immediately issues an unused ID that is randomly generated to the connection and makes an entry in the routing table. This is done in order to record where packets are to be forwarded and the amount of bandwidth allocated to it.

4. A message relating to the connection setup with the IDs of the nodes that form the outgoing path and the QoS constraints for both directions is then sent along the path to the server. This is done in order to reserve and allocate bandwidth for this connection.

5. The intermediate routers are nodes, other than the client and server routers, in an outgoing or incoming path. When a connection setup message arrives, the intermediate node gets the ID and the bandwidth required for the next node.

6. If there is enough bandwidth along the outgoing path to the next node, an entry is made in the routing table. Also, the appropriate delay and the cost of the connection setup message are updated, and the message is forwarded. Otherwise go to step 1.
7. When a connection setup message is received by the server, the previous node makes an entry in the routing table with the server as the next node in the path.

8. Next, the server performs a source based search for a path that meets the incoming bandwidth availability. If successful, it issues an ID that is randomly generated to the connection going in the opposite direction. A routing table entry is done for the second time. If not successful, inform the client and stop connection process.

9. A connection setup message with the incoming path, total delay and cost information present in the old message is sent back to the client router.

10. If enough bandwidth is available, the client router finally receives the connection setup message generated by the server router. If enough bandwidth is not available, then CR (Client Router) tries to find two paths by different links to the server.

11. This indicates that both the client side and the server side connections are ready and that the client can be informed of the delay and cost totals.

### 7.6 Source QoS Routing: Advantages and Disadvantages

The source routing is simple in a way that it transforms a distributed problem into a centralized one. The methodology of source routing is that the source node calculates the entire path locally by maintaining a complete reserved global state. This actually avoids the distributed computing problems such as deadlock detection problem. Many source routing algorithms are very simple and easy to implement, evaluate, debug and upgrade.

The source routing has some disadvantages also. The global state maintained at every node has to be updated frequently. This is done to adjust to the dynamics of network.
parameters such as bandwidth and delay. It makes the communication overhead to a great extent for relatively large scale networks. The computation overhead at the source is also excessively high. This is the problem in the case of multicast routing or when multiple constraints are involved. Finally, the source routing has the scalability problem. It is not practical for any single node to have access to the detailed state information about all nodes and all links in a large network [107].

7.7 Heuristic QoS Routing Algorithm

The proposed Heuristic QoS Routing algorithm uses Label based probing method. A short note on probes is given below before proposing the algorithm for Heuristic QoS Routing.

Probes are nothing but routing messages. The basic idea of probing is as follows. Several probes are sent from the start node to the destination node. Each probe is forwarded independently according to some QoS metrics such as delay, throughput etc. At the destination node, every received probe represents a possible route. From the set of the received probes, the one that satisfies the given QoS constraints is selected. Furthermore, since multiple probes are used for one routing request, more than one path can be explored.

The flowchart for the proposed Heuristic QoS Routing method is given in Fig. 7.1.
Stan

I

Client sends a Label Request Message to the Sener

Sener issues “n” Labels and sends probes with labels to the Client. Yellow Labels = Paths with smaller delays. Green Labels = Paths with smaller costs.

Nodes receive a probe with “n” labels.

Node makes “n” copies of probes

Node distributes the labels to new probes

Node forwards the new probes to Clients

The Labels are not validated and the Probe is sent to the client.

If the path of the probe has enough B/W

Yes

No

Probe record the path, total cost and total delay

Connection Setup message is sent to the server

Server sends an acknowledgement message to the Client.

Total Cost and Total Delay is informed to the Client

Stop

Figure 7.1 Flowchart for Heuristic QoS Routing Algorithm
The algorithm followed in the Client Server model for Heuristic QoS Routing is as follows.

1. A request message is sent to the server when a connection request arrives at the client.

2. When the server receives the request message, it issues a certain number of "n" labels and sends probes, containing the labels, to the client router again.

3. In order to search for good paths in terms of delay and cost, y labels are colored yellow and g labels are colored green. Yellow labels prefer paths with smaller delays, while green labels prefer paths with lower costs.

4. Whenever a node receives a probe pr with N(pr) labels, it makes at most N(pr) copies of pr. It also distributes the labels among the new probes, and forwards them along the selected outgoing paths towards the client.

5. The probes can travel only along the paths that satisfy the bandwidth constraint. If a probe arrives at a node whose outgoing paths cannot give the requested bandwidth, its label(s) are not validated, and the probe is sent to the client.

6. Each probe records the path, total cost, and total delay of the path it explores. This is done since it can choose the best valid path when the client receives all labels.

7. Once a path has been selected, a connection setup message is sent to the server. The message sent in Heuristic Routing is different from those used in Source Routing in the sense that it reserves and allocates bandwidth for an incoming connection, and at the same time the message travels in the outgoing path also.
8. When the connection setup message is received by the server, it replies with an acknowledgment message depending on the result.

9. Finally, when the client receives the acknowledgement for connection, the total delay and cost of the incoming path in the client is informed.

7.8 Heuristic QoS Routing Algorithm: Advantages and Disadvantages

In this routing, the computation of paths is in a distributed manner among the intermediate nodes between the source and the destination. This makes the routing response time shorter and makes the algorithm more scalable. The searching of multiple paths in parallel for a feasible one is made in this algorithm, which increases the chance of success. Most of the existing Heuristic routing algorithms require each node to maintain a global network state based on which the routing decision is made on a hop by hop manner.

The distributed routing algorithms almost have similar disadvantages as Source Routing algorithms. In addition, when the global states at different nodes are inconsistent, loops may occur. A loop can be detected easily when the routing message is received by a node for the second time.

7.9 Experiments and Evaluations

A Client Server program is written in VC++ [108][109] for Source and Heuristic QoS routing Algorithms. The program has been designed in such a manner that it is very easy to use and at the same time very efficient in computing the three metrics namely message overhead, response time and path delay for the comparison of the two algorithms. The reason for writing a Client Server program instead of using the available simulators is because, some of the simulators such as “Insignia”, “Rice Monarch Project implementation
of Source Routing” or Open Source Software’s like “Quaggra”, “XORP” or “BIRD” provide implementations for Routing algorithms in which the experimentations for the three metrics that is considered in this work is difficult. Another difficulty with the existing software’s is that, they are executed in various computer platforms which are not easily available.

The inputs to the program are the Topology, the Algorithm to be used, Source, Destination, Number of requests and Bandwidth. The source is assumed as the Client and the destination as the Server. The connections are made based on the network topology as given in Fig. 7.2.

![Figure 7.2 The Network Topology](image)

During processing, when the number of requests and the bandwidth are given as input, the three metrics are evaluated by the program based on the following definitions.

1. **Average message overhead** = \( \frac{\text{rms}}{\text{cr}} \)

   where \( \text{rms} \) is the total number of routing messages sent and \( \text{cr} \) is the total number of connection requests.

2. **Average response time** = \( \frac{\text{tr}}{\text{cr}} \)
where \( tr \) is the time to handle all requests (accept or reject) and \( cr \) is the total number of connection requests.

3. Average path delay = \( \frac{tcp}{ecp} \)

where \( tcp \) is the total cost of all established connection paths and \( ecp \) is the number of established connection paths.

The Outputs generated by the program are computed results of Response Time, Message Overhead and Path Delay.

The Block diagrams shown in Fig. 7.3 and Fig. 7.4 give the working of the Client Server program for Source and Heuristic Algorithms.

Figure 7.3 Block Diagram for Source QoS Routing (Client Server Program)

The various functions of the block diagram are given below for the Source QoS Routing.

1. **Input**

   The inputs to the program are the topology, the algorithm to be used, source, destination, number of requests and bandwidth.
2. **CFindPath**
   In this function, the Client Router finds paths by different links to the Server. It also checks for paths and stops the connection process if the path is not found.

3. **CUnusedIDIss**
   This function is used to issue an unused ID for the Client, to make an entry in the Routing Table, send node ID to Source Router.

4. **BWCheck**
   This function is used to check if there is enough Bandwidth.

5. **CRTableEnt**
   The Client Router table entry, the cost and delay updating are done by this function.

6. **SFindPath**
   The Source Router finds paths by different links to the Client, checks for paths and stops the connection process if the path is not found in this function.

7. **SUnusedIDIss**
   This function is used to issue an unused ID for the Server, to make an entry in the Routing Table, send node ID to Client Router.

8. **SrTableEnt**
   The Source Router table entry, the cost and delay updating are done by this function.
9. **ConRec**

In this function the Client Router is used to receive the connection set up message send by the Source Router.

10. **ConEsd**

This function is used to establish the connections and to inform the delay and cost totals to the Client.

11. **Output**

The outputs generated by the program are computed results of Response Time, Message Overhead and Path Delay.

![Figure 7.4 Block Diagram for Heuristic QoS Routing (Client Server Program)](image)

The various functions of the block diagram are given below for the proposed QoS Heuristic Routing.
1. **Input**

The inputs to the program are the topology, the algorithm to be used, source, destination, number of requests and bandwidth.

2. **SLabelReq**

This function is called when the router receives a connection request from a client wishing to use distributed routing. It sends a message to the destination router asking it to generate labels and send probes.

3. **ProLabelReq**

This function handle label request message from the source router of a connection.

4. **SProbe**

This function performs the following operations: Calculate message length, insert message length, copy probe structure to message, find index of receiver in the global state list and send probe.

5. **ProProbe**

This function makes n copies of the probes and distributes the labels to new probes.

6. **GProbes**

The ProLabelReq function generates probes that are sent across the network attempting to find a feasible path to a destination router. GProbes receives all of these probes at the destination router. Once all probes have arrived, the best path in terms of delay is chosen. Invalid probes are ignored. If at least one path is valid, a
connection setup message is sent back to the source allocating resources along the way.

7. **RejLabelReq**

   This function informs the client router that no labels will be generated.

8. **PZeroLabels**

   This function informs the client that the connection failed due to insufficient resources.

9. **InvLabels**

   This function is executed at the intermediate nodes if resources are not available and the label is invalidated.

10. **SLabelConSetup**

    This function reserves resources along the path received in one of the valid probes. Allocation must be done backward until the generator of the probes is reached.

11. **CLabelConSetup**

    This function reserves resources following the best of the valid paths. This is done backward from the router that received the probes to the router that generated them.

12. **PLabelConAck**

    This function informs the client that a connection with the server has been established.
13. **RConID**

After all probes are received, the connection setup (allocation of resources and update of routing tables) phase begins. If for some reason, this process fails, the router to which the server is connected must be informed so it can release the connection ID it reserved when it generated the probes.

14. **Output**

The outputs generated by the program are Response Time, Message Overhead and Path Delay.

Fig. 7.5 shows the screenshot for the QoS based Source Routing Algorithm. This figure depicts the screenshot of the client server program when the connection requests are given as 10 and the bandwidth is given as 256. The values of the response time, message overhead and the path delay are computed and are displayed.
The Fig. 7.6 shows the screenshot of the client server program developed for the QoS based Source QoS Routing Algorithm for 30 requests. In the program developed, when the connection request is given as 30 and the bandwidth is given as 512 respectively, the response time, the message overhead and the path delay are computed and are shown.

Figure 7.6 The Screenshot of QoS based Source Routing for 30 requests.

Figure 7.7 The Screenshot for QoS based Heuristic Algorithm for 20 requests.
The Fig. 7.7 shows the working of the client server program developed for the QoS based Heuristic QoS Routing Algorithm for 20 connections. When the connection request is given as 20 and the bandwidth is given as 128 respectively, the response time, the message overhead and the path delay are computed and are shown.

Fig. 7.8 displays the computed result of response time, message overhead and path delay when the connection request is given as 30 and the bandwidth is given as 1024 for the QoS based Heuristic Routing Algorithm.

Figure 7.8 The Screenshot for QoS based Heuristic Algorithm for 30 requests
7.10 Discussions:

1. **Response Time** (The time taken to accept or reject a connection)

   The Table 7.1 shows the Response Time for 10, 20 and 30 connection requests in Heuristic and Source Routing for various bandwidths.

   **Table 7.1 Response Time**

<table>
<thead>
<tr>
<th>Average Bandwidth (kbps)</th>
<th>10</th>
<th>20</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Heuristic</td>
<td>Source Heuristic</td>
<td>Source Heuristic</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>4</td>
<td>32</td>
<td>2</td>
</tr>
<tr>
<td>128</td>
<td>14</td>
<td>174</td>
<td>32</td>
</tr>
<tr>
<td>256</td>
<td>33</td>
<td>202</td>
<td>31</td>
</tr>
<tr>
<td>512</td>
<td>20</td>
<td>248</td>
<td>22</td>
</tr>
<tr>
<td>1024</td>
<td>19</td>
<td>220</td>
<td>25</td>
</tr>
</tbody>
</table>

   The graphical representations given below shows the performance of Source and Heuristic QoS Routing for Response Time, Message Overhead and Path Delay with Connection Requests 10, 20 and 30 and for bandwidths ranging from 64 Kbps to 1024 Kbps. S10, S20, S30 denotes the Connection Requests for Source Routing and H10, H20, H30 denotes the Connection Requests for Heuristic Routing.

   Fig. 7.9 is the graphical representation for Table 7.1 which shows that the response time for the Heuristic QoS Routing is more than the Source QoS Routing.
From Table 7.1, two situations are found when the Heuristic and Source based algorithms are used with respect to the response time. The first situation is that when the bandwidth is increased, the response time shoots up initially when the bandwidth is increased from 64 Kbps to 128 Kbps. The second observation is that as when the bandwidth is increased further, it is seen that there is only a slight variation in the response time for Source and Heuristic Routing.

In Fig. 7.9 it is seen that the Response Time is always higher for Heuristic Routing than Source Routing for all Connection Requests and varying Bandwidths. There are two reasons that account for the disparity between Heuristic Routing and Source Routing response times. The first one is that whenever the bandwidth constraint is hard to meet due to low residual bandwidths at the links, the Source Routing search usually fails at the client router and the connection is immediately rejected without incurring any transmission time penalty. This is not the case with Heuristic Routing because it forces a label request...
message to travel to the server router and some kind of response to return to the client router. The second reason that affects Heuristic Routing’s response time is that in order to achieve a high success ratio, probes carrying labels may have to travel along intricate paths and take longer times to reach the client router. Furthermore, the client router must wait until all labels have been received before it can select the lowest delay path for the connection.

2. **Message Overhead** (Number of messages)

The Table 7.2 shows the Message Overhead for 10, 20 and 30 connection requests in Heuristic and Source Routing for various bandwidths.

| Average Bandwidth (kbps) | Connection Requests | | | |
|---|---|---|---|
| | 10 | 20 | 30 |
| Source Heuristic | Source Heuristic | Source Heuristic | Source Heuristic |
| 64 | 12 | 17 | 4 | 9 | 2 | 4 |
| 128 | 13 | 24 | 7 | 17 | 4 | 10 |
| 256 | 16 | 26 | 8 | 16 | 5 | 11 |
| 512 | 16 | 28 | 7 | 17 | 5 | 13 |
| 1024 | 16 | 27 | 6 | 21 | 4 | 15 |

Figure 7.10 shows the graphical representation of Table 7.2.
From Fig. 7.10, it is found that the Message Overhead of Source based routing is always lower than Heuristic Routing. This is due to two reasons. First, the labels may travel along multiple paths instead of the single path produced by the Source Routing algorithm. Second, whenever the Source Routing fails at the client router, no message overhead is incurred.

3. **Path Delay (Total cost of all established connection paths)**

The Table 7.3 shows the Path Delay for 10, 20 and 30 connection requests in Heuristic and Source Routing for various bandwidths.

Table 7.3 shows that whenever Source Routing manages to find a path that satisfies the bandwidth requirement, it produces paths of slightly lower delay than the Heuristic Routing algorithm.
The readings in Table 7.3 show that there is a sudden increase in average path delay when the average bandwidth is increased from 64 to 128 kbps. The main reason for this is that, when the bandwidth request is set to 64 kbps, only a few requests are accepted initially, but when the average bandwidth is set to 128 kbps, there are more path
possibilities, which results in an increased response time and path delay. For 1024 kbps, there is an increase in the average path delay. The Graphical representation with respect to Path Delay for the complete readings found in Table 7.3 is given in Fig 7.11.

7.11 Results and Findings

It is found that the Source Routing performs better than the Heuristic Routing with respect to the three metrics. This is because whenever a packet arrives at a switching device for Source Routing, the decision to forward it to the next node is not needed since they are all stored in the packet itself. The node looks only at the path information header in the packet in order to determine the port on which it should forward the packet.

Source Routing is a method of transferring a packet through a network in which the path is already determined by the source whereas in Heuristic Routing, it is sufficient for the source to know only about how to get to the next node, and so on until the destination is reached. Heuristic Routing has only the address of the destination rather than a complete specification of the route by the source node. Source Routing uses a path that is already determined by the source. If there is any problem in packet transfer, then the proposed Heuristic Routing is more dependable than Source Routing. Heuristic Routing finds an optimal path using label based probing method, which Source Routing does not use since the path is predefined.

7.12 Conclusion

This chapter analyses two QoS based Routing Algorithm namely the Source based Routing and the proposed Heuristic based Routing. The advantages and the disadvantages
of the two routing algorithms are also discussed. Moreover a Client Server based model has been developed using VC++ for computing three important metrics such as the Response Time, Message Overhead and Path Delay. These metrics are computed whenever the connection requests and the bandwidth values are given by the user in the developed model. The readings are tabulated, the graphs are drawn and discussions regarding the performance of the two algorithms for the three metrics are done. It is found that the Source Routing performs better than the Heuristic Routing with respect to the three metrics. As a future work, the client server model that has been developed can be used for computing more metrics such as QoS Ratio and Mobility Ratio.