Quality of Service Routing: A Multi-Path Approach

A. Dr. Shuchita Upadhyaya, B. Gaytri Devi

1Department of Computer Science and Applications, Kurukshetra University, Kurukshetra
shuchita_bhasin@yahoo.com

2GVM Institute of Technology and Management, DCRUST, Sonipat
gayatri.dhingra1@gmail.com

Abstract
The demand of real time multimedia applications have been increased in today’s scenario of Internet. To fulfil this requirement, Quality of Service (QoS) factors have become necessary to be present in the network. For example, transmission of video over a computer network should be without undesirable delay, jitter and a medical image or a robot control packet may be required to be transmitted over a network with the minimum end-to-end delay. The present Internet routing mechanisms (based on the best-effort paradigm) are unlikely to provide such end to end performance guarantees requisite in these applications. Here is a need of the mechanism which will consider these factors (delay, jitter, bandwidth etc.) for the transmission. One of the components of that mechanism is QoS routing. Multipath approach can be merged into QoS Routing to catch its maximum advantage as in some situation a single path is not able to fulfil all the QoS requirements. The benefits of provisioning multiple QoS paths are reliable QoS support and uniformly balanced network load.

This paper describes an approach based on identifying multiple paths to provide QoS routing. A source routing algorithm has been presented in this paper. The algorithm is an extension to the algorithm proposed by Wang & Crowcroft (Wang et al, 1996). The algorithm proposed by (Wang et al, 1996) finds a single path which satisfies QoS parameters-residual bandwidth and propagation delay and finds a bandwidth-delay constrained path. The algorithm first eliminates all the links that do not meet the bandwidth requirement. Then it finds the minimum delay path from source to destination using Dijkstra algorithm. A multipath extension to this algorithm i.e. M-Bandwidth-Delay constrained algorithm has been proposed in this paper. The algorithm has been implemented in MATLAB and the results obtained have also been presented.

Keywords: Multipath routing, multiple paths, Single shortest path, Quality of Service, Bandwidth, Delay.
I. INTRODUCTION

Quality of Service (QoS) puts some restrictions in the form of certain constraints on the path. These constraints may be desired bandwidth, delay, variation in delay experienced by receiver (jitter), packet loss that can be tolerated, number of hops, cost of links etc.

The fundamental problem of routing in a network that provides QoS guarantee is to find a path between specified source and destination node pair that simultaneously satisfies multiple QoS parameters.

These parameters are represented in the form of metrics. One metric for each constraint is to be specified like bandwidth metric, jitter (variation in delay) metric, delay metric, number of hops metric, packet loss ratio etc. from one node to all other nodes in the network. Metric for a complete path with respect to each parameter is determined by the composition rules of metrics. The three basic rules are (Xiao et al, 1999).

(i) Additive Metric: The value of the constraint over the entire path is the addition of all links constituting path. For Example- delay, hop count, cost or jitter.

(ii) Multiplicative Metric: Using this metric, the value for the complete path is multiplication of metric value of all its edges. Examples are – reliability (1-lossratio) and error free Transmission (probability)

Multiplicative metric can be converted into additive by taking logarithm.

(iii) Concave Metric: In this metric, either min edge value or max edge value is taken as constraint value for a path among all the edges of that path. Such as Bandwidth. For a complete path, the constraints may be required either as a constrained form or in an optimization form. In constrained form, some condition is put on constraint value e.g. Choose that path only which has delay less than or equal to 60 ms. The path obeying the condition is called feasible. On the other hand optimization refers to path having minimum or maximum value for a constraint e.g. Choose the path that has minimum delay among all the paths. This path is called optimal path.

The further QoS issues have been discussed in (Upadhaya et al, 2010)(Upadhaya et al, 2010).

To provide user- or application-level Quality of Service (QoS) guarantee multipath routing strategy can be used for the transmission of QoS sensitive traffic over the network. Multipath routing means using multiple paths instead of using single path to forward the traffic. If multiple paths are being used for the transmission of the traffic then the traffic will be redirected to the backup path if active path fails. In this way robustness can be achieved. On the other hand load balancing for communication network to avoid network congestion & optimize network throughput also requires multi paths to distribute flows. Robustness & load balancing are aspects of QoS routing. So multipath can be proved very valuable for Quality of service.
For multiple paths, there are two characteristics to address: path quantity and path independence. Path quantity means number of paths and path independence refers to disjoint paths. Disjoint paths may be either link-disjoint or node-disjoint. A pair of paths is link-disjoint if, they have no common (i.e. overlapping) links. A pair of paths is node-disjoint if, they have no common nodes. Node-disjoint paths provide more reliability than link-disjoint paths. In robust communication networks, a connection usually consists of link- or node-disjoint paths: one active path, and other back-up paths. The others multi-path issues have been discussed in (Upadhaya et al, 2011). Node-disjoint paths provide more reliability than the link disjoint paths.

In general a link-disjoint paths algorithm can be extended to a node-disjoint algorithm with the concept of node splitting, i.e. replacing one node with two nodes that are linked together via a link with zero-valued weights. (Masip-Bruina et al , 2006)

To determine the multiple paths various k-shortest path algorithms have been used. The k-shortest means determining not only the shortest, but also the second, the third ……………the kth shortest path. To find the k-shortest path, shortest path algorithms Dijkstra & Bellman-ford algorithms can be used in the generalization form (Martins et al, 1998). The K-shortest paths are limited to defining alternate paths without consideration of QoS constraint.

So here is a need of enhancing the above mentioned algorithms to consider QoS parameters. Various domains that have implemented multipath & Quality of service collectively presented in literature have been investigated in (Upadhaya et al, 2011). In this paper we have chosen the Dijkstra algorithm for implementation.

Prominently, there are two routing strategies: source routing and distributed routing. They are classified according to how the state information is maintained and how the search of feasible paths is carried on. To provide Quality of Service (QoS) guarantee both of the routing schemes –source routing and distributed routing can be used. In source routing, the path computation is done at source node whereas in distributed routing, the path computation is distributed among intermediate routers between source and destination. Both source routing and distributed routing have important roles to play in QoS routing. Today distributed routing is the common strategy for routing in the Internet while source routing is used in today’s Internet for special cases, such as mapping the network with trace route, troubleshooting, or forcing an alternate link to traffic flow to avoid congestion.

One source routing algorithm proposed by Wang and Crowcroft in (Wang et al, 1996) identifies single QoS path. In this paper, we have presented multi-path extension to this algorithm i.e. M-Bandwidth-Delay constrained algorithm for finding multiple paths that satisfy QoS criteria. The layout of the paper is as follows: Section 2 discusses the proposed approach. Section 3 presents output in MATLAB and Section 4 concludes the paper.
II. IMPLEMENTATION OF QUALITY OF SERVICE USING MULTI-PATH

The proposed algorithm is based on (Wang et al, 1996). The two metrics considered in (Wang et al, 1996) are residual bandwidth and propagation delay. In (Wang et al, 1996), the algorithm finds bandwidth-delay constrained path. As bandwidth is a concave metric and delay is an additive metric, so it finds the path whose each edge bandwidth satisfies the bandwidth requirement and whose total delay (addition of delay of all edges) should be less than the maximum delay allowed. The algorithm first eliminates all the links that do not meet the bandwidth requirement. Then, it finds the minimum delay path from source to destination using Dijkstra algorithm. The algorithm is terminated when either destination is reached or delay exceeds the threshold.

The algorithm presented in (Wang et al, 1996) finds the single best path. We have extended this algorithm to find multiple paths instead of single path. Our objective is to find link disjoint paths which satisfy the QoS constraints. To find disjoint paths ‘Find and Removal’ method have been used. The basic theme of the algorithm is that, firstly, it eliminates all the links that do not meet the bandwidth requirement. Then, it finds a (the best one) path between s(source) and t(destination) satisfying delay. Once a path is found, all the links in that path are removed from the network graph and the algorithm is run on the pruned graph to find the next minimum-delay path. The process is continued until the algorithm comes up with the best m number of shortest paths from s to t or delay exceeds the delay threshold or the network is disconnected from source to destination. Here the value of m is number of adjacent nodes of source. Thus, the algorithm finds multiple link disjoint loop-less paths from s to t satisfying QoS criteria- residual bandwidth, delay in the increasing order of delay.

A. Data Structure

For the construction of paths, we use certain parameters. The following are the list of parameters used in developing the path construction algorithm.

- Node structure holds its status (tentative, permanent), its total delay from source, its predecessor, its bandwidth and number of hops from source i.e.
  - node. status = Shows the status of node= "permanent" or "tentative."
  - node. delay= Total delay of path from the source to that node.
  - node. pred = Its previous node according to path found.
  - node. bandwidth= Value of minimum bandwidth edge from source to that node
  - node. hops=Number of hops from source to that node.
- $b[i,j]$ =Residual bandwidth for each edge.
- $d[i,j]$ = Delay value for each edge.
- $band\_th$ =Bandwidth threshold/requirement (Path should have bandwidth equal or greater than $band\_th$)
B. M-Bandwidth-Delay constrained algorithm

The steps of M-Bandwidth-Delay constrained algorithm are-

Step 1: Set $d_{ij} = 0$ & $b_{ij} = 0$, if $b_{ij} < \text{Band\_th}$. (Eliminates all links that do not meet the bandwidth requirement by setting their metric matrix to 0.), count = 0.

Step 2: Initialize node[s].pred=0, node[s].delay=0, node[s].status=permanent, node[s].bandwidth=$\infty$.
node[i].pred=s, node[i].delay=$\infty$, node[i].bandwidth=$\infty$, node[i].status=tentative, for all $i \neq s$. k=s

Step 3: for all i whose node[i].flag=tentative && ($d_{ki} > 0$) 
If (node[i].delay > $d_{ki} + \text{node[k].delay}$) begin
    node[i].delay = $d_{ki} + \text{node[k].delay}$
    node[i].bandwidth = min(node[k].bandwidth, $b_{ki}$)
    node[i].pred = k
    node[i].hops = node[k].hops + 1
end
End if
Step 4 Check for connectivity of source & destination if false then exit.
Step 5: Change k so that node[k].delay = Min (node[i].delay) (Here i is tentative node)
node[k].flag=permanent
Step 6 : Check 
If k = t then
begin
If (node[k].delay <= delay_th) then
begin
    count = count + 1
    record the path by following predecessor link.
    If (count <= m) then 
        Set bij, dij=0 for all edges available in path found & Goto step 2 
    Else
    Stop
End if
End
Else
   Stop
End if
End
Else
   Go to Step 3
End if
Yes

Among the tentative nodes, find the node having least delay value & make it as active node. Mark active as permanent.

active = t

delay ≤ delay_th

Yes

count = count + 1, Record the path by tracing the predecessor node. Subtract every entry in b[i,j] and d[i,j] that appears in the path.

count = m

Yes

Stop

No

Check Connectivity

Yes

No

count = m

Stop

No

Sto

Sto

Start

Eliminate all links that do not meet the bandwidth requirement by making its entry in b[i,j] and d[i,j] as zero. count = 0

Identify source and destination node as s and t. Initialize the parameters of the node structure record (for all nodes) and set their status to "tentative" and their delay, bandwidth to "infinity"

Make source as active node and set its bandwidth to ∞ and delay to 0 & make

Update all tentative nodes and record its structure i.e. its delay, its bandwidth, hops, its predecessor according to active node.

Figure 1. Flowchart of M-Bandwidth-Delay constrained Algorithm
C. Illustration of Algorithm

We have illustrated the algorithm through the following example. Here is a network of 6 nodes. Each edge is represented by its residual bandwidth & delay metric.

Thus bandwidth and delay matrix of Network are-

**Table I**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>
The following parameters have been assumed-
Bandwidth threshold - 4
Delay threshold - 10
Source - 1
Destination – 6
Total number of paths found-Adjacent edges of Source - 3

The problem can be defined as - find multiple disjoint paths between 1 and 6 in the given network. The paths should follow the following conditions-
The bandwidth of the paths should be minimum 4.
The delay of the paths should not exceed 10.

As bandwidth is concave metric, so each of the edges in the path found should have value greater than or equal to 4 and delay is an additive metric, so the sum of the delay of edges constituting the path should be less than or equal to 10.

According to the algorithm, the edges are removed from the graph that does not satisfy the bandwidth requirement by setting their delay value and bandwidth value to 0.

As the bandwidth requirement is 4 so the edges having less than bandwidth 4 should be removed from the graph. Among all the edges of graph, the edge 4-5 is having bandwidth 3 i.e. less than 4. It will not be considered further for path finding.
Now the resultant network will be:

**TABLE II**

**DELAY MATRIX**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>
In this network, Elimination of edges not satisfying bandwidth requirement has been done. Now the bandwidth of all the edges is equal or greater than 4.

i. First Iteration

In the network depicted in fig. 5.3, the shortest delay path from 1 to 6 as computed by the proposed algorithm is

<table>
<thead>
<tr>
<th>Path</th>
<th>Bandwidth of Path</th>
<th>Delay of Path</th>
<th>Hop-count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-3-6</td>
<td>5</td>
<td>7</td>
<td>2</td>
</tr>
</tbody>
</table>

Remove the edges found in this path from the network.

The resultant network will be:
ii. Second Iteration

Now in the network of figure 4, the least delay path from 1 to 6 is

<table>
<thead>
<tr>
<th>Path</th>
<th>Bandwidth of Path</th>
<th>Delay of Path</th>
<th>Hop-count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-4-6</td>
<td>4</td>
<td>8</td>
<td>2</td>
</tr>
</tbody>
</table>

Remove the edges found in this path from the network. The resultant network will be:

Figure 5. Network after removal of edges of second path
iii. Third iteration

Now in network of figure 5, the least delay path from 1 to 6 is

<table>
<thead>
<tr>
<th>Path</th>
<th>Bandwidth of Path</th>
<th>Delay of Path</th>
<th>Hop-count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-2-5-6</td>
<td>4</td>
<td>9</td>
<td>3</td>
</tr>
</tbody>
</table>

Here the value of total number of paths found has reached to 3. So the algorithm will terminate here.

Thus, the multiple disjoint QoS paths found in the given network are as follows:

<table>
<thead>
<tr>
<th>Path</th>
<th>Bandwidth of Path</th>
<th>Delay of Path</th>
<th>Hop-Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-3-6</td>
<td>5</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>1-4-6</td>
<td>4</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>1-2-5-6</td>
<td>4</td>
<td>9</td>
<td>3</td>
</tr>
</tbody>
</table>

Thus the algorithm finds multiple disjoint paths satisfying bandwidth and delay requirements in the order of increasing delay. This algorithm has been implemented in MATLAB. The output obtained in MATLAB has been presented in next section.
III. MATLAB Output

Input number of nodes:-6

Enter bandwidth and delay matrix:-
Enter bandwidth between [1] [1] 0
Enter delay between [1] [1] 0
Enter bandwidth between [1] [2] 5
Enter delay between [1] [2] 3
Enter bandwidth between [1] [3] 5
Enter delay between [1] [3] 2
Enter bandwidth between [1] [4] 4
Enter delay between [1] [4] 5
Enter bandwidth between [1] [5] 0
Enter delay between [1] [5] 0
Enter bandwidth between [1] [6] 0
Enter delay between [1] [6] 0
Enter bandwidth between [2] [1] 5
Enter delay between [2] [1] 3
Enter bandwidth between [2] [2] 0
Enter delay between [2] [2] 0
Enter bandwidth between [2] [3] 0
Enter delay between [2] [3] 0
Enter bandwidth between [2] [4] 4
Enter delay between [2] [4] 2
Enter bandwidth between [2] [5] 4
Enter delay between [2] [5] 3
Enter bandwidth between [2] [6] 0
Enter delay between [2] [6] 0
Enter bandwidth between [3] [1] 5
Enter delay between [3] [1] 2
Enter bandwidth between [3] [2] 0
Enter delay between [3] [2] 0
Enter bandwidth between [3] [3] 0
Enter delay between [3] [3] 0
Enter bandwidth between [3] [4] 5
Enter delay between [3] [4] 4
Enter bandwidth between [3] [5] 0
Enter delay between [3 ][5 ]0
Enter bandwidth between [3] [6] 5
Enter bandwidth between [4] [1] 4
Enter delay between [4 ][1 ]5
Enter bandwidth between [4] [2] 4
Enter delay between [4 ][2 ]2
Enter bandwidth between [4] [3] 5
Enter delay between [4 ][3 ]4
Enter bandwidth between [4] [4] 0
Enter delay between [4 ][4 ]0
Enter bandwidth between [4] [5] 3
Enter delay between [4 ][5 ]3
Enter bandwidth between [4] [6] 5
Enter delay between [4 ][6 ]3
Enter bandwidth between [5] [1] 0
Enter delay between [5 ][1 ]0
Enter bandwidth between [5] [2] 4
Enter delay between [5 ][2 ]4
Enter bandwidth between [5] [3] 0
Enter delay between [5 ][3 ]0
Enter bandwidth between [5] [4] 3
Enter delay between [5 ][4 ]3
Enter bandwidth between [5] [5] 0
Enter delay between [5 ][5 ]0
Enter bandwidth between [5] [6] 5
Enter delay between [5 ][6 ]3
Enter bandwidth between [6] [1] 0
Enter delay between [6 ][1 ]0
Enter bandwidth between [6] [2] 0
Enter delay between [6 ][2 ]0
Enter bandwidth between [6] [3] 5
Enter delay between [6 ][3 ]5
Enter bandwidth between [6] [4] 5
Enter delay between [6 ][4 ]3
Enter bandwidth between [6] [5] 5
Enter delay between [6] [5] 3
Enter bandwidth between [6] [6] 0
Enter delay between [6] [6] 0

The bandwidth matrix is:

\[
\begin{bmatrix}
0 & 5 & 5 & 4 & 0 & 0 \\
5 & 0 & 0 & 4 & 4 & 0 \\
5 & 0 & 0 & 5 & 0 & 5 \\
4 & 4 & 5 & 0 & 3 & 5 \\
0 & 4 & 0 & 3 & 0 & 5 \\
0 & 0 & 5 & 5 & 5 & 0 \\
\end{bmatrix}
\]

The delay matrix is:

\[
\begin{bmatrix}
0 & 3 & 2 & 5 & 0 & 0 \\
3 & 0 & 0 & 2 & 3 & 0 \\
2 & 0 & 0 & 4 & 0 & 5 \\
5 & 2 & 4 & 0 & 3 & 3 \\
0 & 3 & 0 & 3 & 0 & 3 \\
0 & 0 & 5 & 3 & 3 & 0 \\
\end{bmatrix}
\]

Enter the source:-1
Enter the destination:-6
Enter delay threshold:-10
Enter bandwidth threshold:-4

The 1 path between 1 to 6 is 1 - 3 - 6
The total delay of path is- 7
The number of hops - 2
The bandwidth of path - 5

The 2 path between 1 to 6 is 1 - 4 - 6
The total delay of path is- 8
The number of hops - 2
The bandwidth of path - 4

The 3 path between 1 to 6 is 1 - 2 - 5 - 6
The total delay of path is- 9
The number of hops - 3
The bandwidth of path - 4
In this MATLAB output network:
The edge values are representing the delay value.
Node 1, Node 6 are - source, destination.
Red Edges are representing- 1\textsuperscript{st} path
Blue Edges are representing- 2\textsuperscript{nd} path
Green edges are representing- 3\textsuperscript{rd} path

IV. Conclusion and Future Work

(Wang et al, 1996) has described an algorithm based on Dijkstra for single path. A multi-path extension to this algorithm (M-bandwidth delay constrained algorithm) has been proposed in this paper with an illustration through an example network. M-bandwidth delay constrained algorithm calculates multiple link disjoint QoS paths. The QoS parameters that have been considered are bandwidth and delay as Bandwidth and delay reflect the most fundamental characteristics of a path in the network. The Most real time applications require guaranteed bandwidth and bounded delay. The problem considering bandwidth and delay is not NP-complete. The worst case complexity of the algorithm in (Wang et al, 1996) is $O(n^2)$. Thus the complexity of
the proposed algorithm is $O(mn^2)$. Here $n$ represents the number of nodes and $m$ represents the number of paths computed. This is a source routing algorithm. Although the algorithm was able to find multiple QoS paths in a network but it may fail to calculate multiple paths in specific topology. Thus the future work is to overcome that failure.

REFERENCES


