DISTRIBUTED QOS ROUTING FOR MULTIMEDIA TRAFFIC

Venkatesh Sarangan, Donna Ghosh and Raj Acharya

Department of Computer Science and Engineering,
State University of New York at Buffalo, NY 14260
(sarangan,donnag,acharya)@cse.buffalo.edu

ABSTRACT

The goal of QoS routing is to find a loopless path through a network that can satisfy a given set of constraints on parameters such as bandwidth, delay and cost. QoS routing algorithms can be classified into three categories. They are Source routing, Distributed routing and Hierarchical routing. In this paper we propose a distributed QoS routing algorithm that gives high call admission rates in a congested network for constraints on metrics like bandwidth, buffer, and CPU usage with a low message complexity.

1. INTRODUCTION

Resource reservation is a necessity for providing guaranteed end-to-end performance for multimedia applications. However in the present Internet setup, resource reservation is not supported. Also, the data packets of these applications could follow different paths and reach the destination out of order, which is not desirable. Hence, the future networks are likely to provide a connection oriented service for real time applications. The path chosen during the call set-up phase should be capable of meeting the requirements of the multimedia applications. The job of QoS routing is to provide such a path (if it exists) from a source to the given destination. The current research on QoS routing has been mainly in the categories of source and distributed QoS routing. Source routing algorithms have the problems of excessive computational overhead at the source and maintaining an up-to-date global state information at every router in the network. The distributed algorithms (DRAs) are free from these problems. Various DRAs have been proposed in [1], [3], [4], [5], [6].

The DRA proposed by Chen-Nahrstedt in [1] has three phases: probing, acknowledgment and failure handling. The probing phase is essentially the QoS routing and it establishes a tentative path between the source and destination such that the path satisfies the QoS requirements of the connection. A router on receiving a probe, selectively floods it on all the outgoing links which are capable of supporting the QoS requirements, except the one on which the probe arrived. Every router selectively floods the first probe received for a given connection and rejects all duplicates. The probing phase ends once a probe reaches the destination and the path the first probe takes to reach the destination is called the tentative path. No reservations are made in the probing phase. The destination sends an acknowledgment to the first probe it receives and it discards all the duplicate probes. This ensures that only one path is established. In the acknowledgment phase, the destination sends an acknowledgment along the tentative path and resources are reserved along the way. A connection is established when the source gets the acknowledgment.

As the network resources change rapidly, an intermediate node on the tentative path (which would have forwarded the probe earlier) might not have the required resources when it gets an ack for the same connection. In such a case, the failure-handling phase is started. The node that was unable to reserve the resources sends a failure message to the downstream routers to free their resources and the connection request is refused. A connection request could be turned down in one other scenario. A router could receive a probe with certain QoS constraints which none of its outgoing links can support. The router will simply discard the probe without forwarding it. If such a condition occurs in all the paths between a <source, destination> pair, the connection will not be set up. In both the cases, the call is said to be blocked.

In this paper, we propose two strategies for selectively flooding the probes so that the overall resource admission rate is increased. The strategy we propose is for metrics like bandwidth, switch buffer, CPU usage etc., where the minimum value of the metric at each router along a path should satisfy the QoS constraint. Examples of metrics that do not come under this category are delay, jitter, etc..

2. INCREASING CALL ADMISSION RATES

The example in figure 1 gives a better understanding of how the call admission rate could be increased. We have
The idea behind the first approach is that, we account for more connections by discouraging resource fragmentation whereas in the second approach, we are reducing the chances of generating a failure message along the tentative path. In the following section, we introduce the term concave metric and our algorithm. We have assumed that the control messages have a higher priority over the data packets. Hence they will not be dropped in any case, even if the network gets congested and will be serviced immediately at the routers. Hence, the control messages incur a negligible queueing delay at all routers. The only delay they experience in a router is the protocol processing time. Further, assuming a high-speed network, the propagation delay can be neglected.

3. CONCAVE METRIC DRA

3.1. Terminologies

Given a directed graph \( G = (V, E) \), such that each edge \( e_{ij} \in E \) has a weighted concave-vector (to be defined soon) \( \tilde{W} \) assigned to it. Let,

1. \( e_{ij} \) = edge from vertex \( v_i \) to vertex \( v_j \) where \( v_i, v_j \in V \) and \( e_{ij} \in E \).
2. \( m(e_{ij}) \) is the metric for an edge \( e_{ij} \) such that \( 0 \leq m(e_{ij}) \leq MAX \). For example, if we consider bandwidth to be the QoS metric, \( MAX \) could be the link capacity.
3. \( P = (v_s, v_a, \cdots, v_i, v_j, \cdots, v_l, v_t) \) be a path in \( G \) which starts at \( v_s \) and ends at \( v_t \) traversing vertices \( v_i \) and \( v_j \).
4. For a path \( P \), metric \( m \) is concave if \( m(P) = \min\{m(v_s, v_a), \cdots, m(v_i, v_j), \cdots, m(v_l, v_t)\} \).
5. \( \tilde{M}(e_{ij}) = (m^1(e_{ij}), \cdots, m^n(e_{ij})) \) is a vector of concave metrics associated with edge \( e_{ij} \) such that \(\dim(\tilde{M}(e_{ij})) = n \) and \( m^q(e_{ij}) \) is the \( q^{th} \) metric; Also \( 0 \leq m^q(e_{ij}) \leq MAX_q \).
6. \( w^q(e_{ij}) \) is the weight assigned to \( m^q(e_{ij}) \) and \( w^q(e_{ij}) \in [0, 1] \).
7. \( \tilde{W}(e_{ij}) = (w^1(e_{ij}), \cdots, w^n(e_{ij})) \) is the vector of normalized weights assigned to each of the \( n \) concave metrics (CM) of edge \( e_{ij} \) such that \( w^q(e_{ij}) \neq w^r(e_{ij}) \) for all \( q \neq r \).
8. \( N(v_i) \) is the Neighbor set of \( v_i \) where \( v_i \in V \) and is defined as \( \{v_j \in V \mid (v_i, v_j) \in E\} \).
9. Every vertex \( v_i \) maintains the vectors \( \tilde{W}(e_{ij}) \) and \( \tilde{M}(e_{ij}) \) for all edges \( e_{ij} = (v_i, v_j) \) where \( v_j \in N(v_i) \).

The input to the algorithm is a request in the form of a source destination vertex pair \( (v_s, v_t) \) where \( v_s, v_t \in V \) and a connection id \( cid \in \text{Natural numbers} \). Each connection request has a constraint vector \( \tilde{Q} \) of the form, \( \tilde{Q} = (Q_1, Q_2, \cdots, Q_n) \). This \( \tilde{Q} \) is the vector of QoS requirements. The output of the algorithm is a path \( P = (v_s, \cdots, v_i, v_j, \cdots, v_t) \) such that \( Q_r \leq m^q(e_{ij}), \forall r \in \{1, 2, \cdots, n\} \) and \( e_{ij} \in P \). The goal is to have a strategy for computing the path for a given connection such that maximum number of connections
could be established in the network.

In the probing phase, every intermediate router forwards the probe on every outgoing link (provided the QoS requirements are satisfied by the link) except on the link on which the probe arrived. The strategy is, to delay the forwarding of the probe on a link $e_{ij}$ for a time $\text{delay}(e_{ij})$ whose value depends on the approach that we choose.

\begin{equation}
\text{Approach - 1 : } \text{delay}(e_{ij}) = K' \times \tau \times df \text{f} \\
\text{Approach - 2 : } \text{delay}(e_{ij}) = K' \times \tau \times df \text{f}'
\end{equation}

where,

\begin{equation}
df \text{f} = \frac{1}{n} \sum_{i=1}^{n} m^{r}(e_{ij}) - Q_r \times \text{MAX}_{r} \times w^{r}(e_{ij})
\end{equation}

and $df \text{f}' = \frac{1}{\text{MAX}_{r} \times \tau}$ squeezed into the interval $[0, 1]$; where $\tau$ is the maximum time taken for a probe to travel one hop and $0 \leq K' \leq 1$. Depending on the network, $\tau$ could be estimated and $K'$ is a scaling factor used to scale down the extra delay introduced at every router by approaches (1) and (2).

Approach-1 (closest-fit heuristic) corresponds to the case where we try to make the probes travel faster through links whose metrics match the QoS requirements closely. Approach-2 (widest-fit heuristic) corresponds to the case where we try to make probes travel faster through links that could support much heavier QoS constraints. At node $v_j$, $p_{ej}(cid)$ refers to the node from which $v_j$ got the first probe; $n_{ej}(cid)$ refers to the next downstream node in the tentative path. A probe is denoted by a tuple as $[k, \bar{Q}, S, T, cid]$. This represents a probe forwarded by router $k$ for connection $cid$ whose source, destination and QoS requirements are $S, T,$ and $\bar{Q}$ respectively. In the next sub-section, we describe an algorithm for increasing the total network resource admission rate.

3.2. CM-DRA (at node $v_i$)

begin

\text{while true, do}

\text{block until receiving a message}

\text{switch (the received message)}

\text{case(1) : probe } [v_k, \bar{Q}, v_s, v_t, cid] \text{ if } (v_i \neq v_k \text{ and } v_i \text{ has not forwarded a probe for the same } cid) \text{ then}

\text{ } p_{vi}(cid) = v_k;

\text{for every node } v_j \in N(v_i), \text{ do}

\text{if } (Q_r \leq m^{r}(e_{ij}), \forall r \in \{1, \cdots, n\}), \text{ then}

\text{delay the probe by using one of the delay conditions; after delay,}

\text{if } (Q_r \leq m^{r}(e_{ij}), \forall r \in \{1, \cdots, n\}), \text{ then}

\text{forward a probe } [v_i, \bar{Q}, v_s, v_t, cid] \text{ on edge } e_{ij}

\text{end if}

\text{end if}

\text{else if } (v_i = v_k) \text{ and this is the first probe } [cid] \text{ received by } v_i \text{ then}

\text{send } v_k \text{ and } \text{ack}[v_i, \bar{Q}, v_s, v_t, cid];

\text{else}

\text{discard the probe;}

\text{end if}

\text{case(2) : ack } [v_k, \bar{Q}, v_s, v_t, cid] \text{ if } (Q_r \leq m^{r}(e_{ij}), \forall r \in \{1, \cdots, n\}) \text{ then}

\text{reserve the metrics } \bar{Q} = (Q_1, \cdots, Q_n) \text{ on link } e_{ik} = (v_i, v_k) \text{ for connection } cid;

\text{n}_{vi}(cid) = v_k;

\text{if } (v_i \neq v_k), \text{ then}

\text{send } p_{vi}(cid) \text{ an ack } [v_i, \bar{Q}, v_s, v_t, cid]

\text{else}

\text{the connection has been successfully established}

\text{end if}

\text{else}

\text{send } v_k \text{ a failure message } [v_j, \bar{Q}, v_s, v_t, cid]

\text{end if}

\text{case(3) : failure } [v_k, \bar{Q}, v_s, v_t, cid] \text{ if } (v_i \neq v_k), \text{ then}

\text{release the resources reserved on link } (v_i, n_{vi}(cid)) \text{ for connection } cid;

\text{send } n_{vi}(cid) \text{ a failure } [v_i, \bar{Q}, v_s, v_t, cid]

\text{end if}

\text{end switch}

\text{end while}

end

3.3. Time and Message Complexity

One of the important issues in distributed algorithms is that, the path returned by the algorithm should be loop free. In any routing algorithm, loops will occur only if a router forwards the same packet more than once. In our algorithm, each probe is identified by its $cid$. The routers forward only the first probe for each $cid$. All the duplicate probes are rejected. This makes sure that no loops are created. During the probing phase, at most one probe travels through each link for a given $cid$. The acknowledgments and failure messages are sent only along the tentative path. So, the total message complexity of establishing a connection is $O(e + 2l)$ where $e$ is the number of edges in the network and $l$ is the length of the longest tentative path. This is same as that of the algorithm given in [1].

By setting $K' = 0$, the CM-DRA algorithm changes to the algorithm given in [1]. If $K' = 0$, the algorithm takes one round-trip time to establish a connection. Under normal conditions, if we assume that the maximum time taken for a
probe to travel one hop is 1, i.e. $\tau = 1$, then the call set-up time is $O(2l)$ units of time, where $l$ is the length of the longest tentative path. By setting $K' > 0$, the CM-DRA algorithm introduces an extra delay for the probes at every hop by an amount of at most $K'$ units. It can be shown that the time taken for a connection establishment in either of the approaches is $O(2l) + O(K' l)$. By keeping $K'$ small, the call set-up time can be reduced. Also, our experiments show that the value of $K'$ does not affect the bandwidth admitted by the network in both the approaches, as long as $K' > 0$.

4. EXPERIMENTAL RESULTS AND DISCUSSION

For any QoS routing algorithm, important performance parameters are the computational overhead, message overhead and the bandwidth/resource admission ratio. For a distributed QoS routing algorithm, assuming that the computational burden is within acceptable limits, the focus is primarily on the message overhead and the bandwidth admission ratio. Bandwidth (resource) admission ratio is defined as the ratio of total bandwidth/resource accepted into the network to the total bandwidth (resource) requested. In the previous section, it was shown that the message overhead in our algorithm is exactly the same as that of [1]. In this section, using experimental results we shall show that the bandwidth (resource) admission ratio of our algorithm is much better than that of [1]. The simulations were done using OPNET, a commercial network simulation package. The comparisons were made on different network topologies. Due to lack of space, we shall show the results of our algorithm on a typical ISP network.

NET-1 shown in figure 2 represents the topology of a typical ISP [2]. Each link is duplex and has a capacity of 155 Mbps (OC-3). The background traffic in each link is set to a value in the range [0, 155 Mbps]. The connection requests arrive at the nodes as per a Poisson distribution. We shall first consider QoS requests with only one metric - bandwidth. The bandwidth requests are uniformly distributed between 64 Kbps and 1.5 Mbps. Results are shown for a connection duration of ten seconds. If the connection duration changes, the bandwidth admitted by both the algorithms changes but the relative performance remains almost the same. Two thousand (2000) connection requests are generated during each simulation run. A $K'$ value of 0.01 was used in the simulations. Multiple simulation runs were done and the table entries are the average of all the runs. Table 1 shows the total bandwidth admitted in the network and the increase in the call set-up time by approach 1 at various load conditions for a single metric. Table 3 shows the same for the multiple metric case. Tables 2 and 4 show the performance of approach 2 for single metric and multiple metrics. The increase in the values are with respect to algorithm suggested in [1]. Constraints of the form $\vec{Q} = [Q_1, Q_2, Q_3]$ are considered. The maximum link capacity for these metrics are respectively $155 \times 10^6$, $200 \times 10^6$, and $50 \times 10^6$.

The results tabulated show that both the approaches increase the total resource admitted into the network. However, this is accompanied by an increased call set-up time. For the closest-fit heuristic, the increase is only around $5 - 6\%$ while for the widest-fit heuristic, the call set-up time increases by at least $30\%$. This could be explained by noting the fact that in the closest-fit heuristic, the delay at each node is proportional to $\text{diff}$ while in approach 2, it is proportional to $1/\text{diff}$. As we are talking about a congested network, the $\text{diff}$ value is usually small. Hence, the closest-fit heuristic introduces a lesser delay than the widest-fit heuristic. In terms of percentage increase in the resource admitted in the network, closest-fit heuristic performs slightly better than widest-fit in the single metric case. For the multiple metric case, no such conclusions could be drawn. Thus, in general, it could be said that closest-fit heuristic is better than widest-fit as it has a smaller call set-up time.

The two heuristics proposed are useful only in a congested scenario. When the network is lightly loaded, there is no necessity for such heuristics, as all the requests will be accepted. Under light load conditions, to avoid the additional delay in the call set-up, the routers could set $K' = 0$. The routers can detect congestion by monitoring the network traffic and use a threshold to determine when the network is congested. To follow any of the heuristics, the routers could use a non-zero $K'$ value.

![Figure 2: NETWORK 1](image)

<table>
<thead>
<tr>
<th>Background Load (Mbps)</th>
<th>% Inc. in Calls Admitted</th>
<th>% Inc. in BW Admitted</th>
<th>% Inc. in Call set-up time</th>
</tr>
</thead>
<tbody>
<tr>
<td>155 - 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>154</td>
<td>6.12</td>
<td>9.23</td>
<td>6.21</td>
</tr>
<tr>
<td>153</td>
<td>6.57</td>
<td>8.65</td>
<td>6.85</td>
</tr>
<tr>
<td>152</td>
<td>6.85</td>
<td>8.16</td>
<td>7.01</td>
</tr>
<tr>
<td>151</td>
<td>6.67</td>
<td>8.34</td>
<td>6.93</td>
</tr>
<tr>
<td>150</td>
<td>5.84</td>
<td>7.10</td>
<td>6.76</td>
</tr>
<tr>
<td>149</td>
<td>6.42</td>
<td>8.86</td>
<td>6.25</td>
</tr>
</tbody>
</table>

Table 1: Performance of Closest-fit heuristic on NET-1 with single metric
5. CONCLUSION

In this paper, we have proposed two heuristics for increasing the total resource admitted into the network by a distributed QoS routing algorithm. The simulation results show that, the increased resource admission rate is accompanied by an increased call set-up time. Approach 1 has a smaller call set-up time than approach 2. The future work lies in developing better heuristics to increase the resource admission rate for multiple constraints.

6. REFERENCES


