Multicast Distributed Routing Algorithm for Providing Network QoS

Teerapat Sanguankotchakorn and Nguyen Thanh Son

Telecommunications Program
School of Advanced Technologies
Asian Institute of Technology
P.O.Box 4, Klong Luang, Pathumthani 12120 Thailand
E-mail: teerapat@ait.ac.th; sonnt3377@yahoo.com

Abstract
Multicast Routing is widely used by various media applications (teleconference, distance learning...). The combination of multicast routing and Quality of Service (QoS) characteristics helps handling the technical requirements and brings better services to end-users.

This paper proposes an extension of Distributed Routing Algorithm (DRA) for supporting multicast routing. The proposed algorithm is called Multicast Distributed Routing Algorithm (MDRA). Our simulations show that MDRA provides better network performance by reducing network overhead while still keeps the call admission ratio in an acceptable range.

Index term – MDRA, Multicast Routing, Quality of Service (QoS), Selective probing.

1. Introduction
Multicast Routing appeared frequently in the recent works. There is large number of problems relating to Multicast Routing and Quality of Service (QoS) constraints ranging from specific multicast algorithms to framework to deal with inaccurate network states or scalability problems. A good survey about QoS Multicast Routing problems can be found in [10]. According to [10], there are three routing strategies: source routing, distributed routing and hierarchical routing. They are classified based on the way the state information is maintained and how the search of feasible paths is carried out.

In the source routing, each node maintains the complete global state, including the network topology and the state information of every link. Based on the global state, a feasible path is locally computed at the source node. A control message is sent out along the selected paths to inform the intermediate nodes of their precedent and successive nodes. In the distributed routing, the paths are calculated in distributed manner. Control messages are exchanged among the nodes and the state information kept at each node is collectively used for the path search. In the hierarchical routing, nodes are clustered into groups, which are further clustered into higher-level groups recursively, creating a multi-level hierarchy. Each kind of routing strategies has its own advantages and disadvantages [1].

The advantage of distributed routing is that the response time can be made shorter and the algorithm is more scalable, since the path computation is distributed among the intermediate nodes between source and destination. Moreover, the distributed routing can, somehow, reduces the inaccuracy in routing, since the computation is not done in one node.

Some typical works about distributed routing are in [2], [5], [6], [8]. In those works, authors proposed heuristic algorithms to solve the problems. However, each work deals with a particular set of parameters. They did not propose a generic framework to apply for all situations (different types of QoS parameters). In [1], authors proposed a generic framework based on selective probing for unicast routing. This framework proposed a general form and more customization could be developed later. The characteristic of this framework makes it to be used widely for many kind of parameters or their combination.

However, the limitation in [1] is that, it did not proposed solution for multicast routing, which is more complicated in term of path determination mechanism (more).

In this paper, we propose an algorithm to work with Multicast Routing with QoS constraints. Our algorithm is a good solution to work with Multicast Routing, since it is also a generic framework and quite flexible. Particular algorithms could be derived by modifying the generic algorithm.

The rest of this paper is organized as follows: section 0 represents Multicast Distributed Routing framework, section 3 is devoted for simulation models and results, and section 4 concludes the paper.

2. Multicast Distributed Routing Framework

2.1 Network Models and Problem Identification

The network is modeled by a graph G (V, E) containing V nodes and E edges. Nodes in the graph represent network routers and edges correspond to communication links between nodes.

Each link has its own parameters (bandwidth, delay, cost etc.), which are used to select paths satisfying QoS requirements.

A multicast operation transmits messages to a group of destinations. A multicast route always takes a tree structure to reduce the number of messages copied in the network, where message copy occurs only at the fork node of the tree. A multicast routing tree can be defined as the following: Given a source node S ∈ V, a set of destination nodes Γ ⊂ V with S /∈ Γ, a routing tree for a multicast connection is a sub-tree of the graph G(V, E) rooted from S that contains all nodes of Γ and an arbitrary subset of V - Γ, whose leaf set consists only of a subset of nodes in Γ.

Nguyen Thanh Son

School of Advanced Technologies
Asian Institute of Technology
P.O.Box 4, Klong Luang, Pathumthani 12120 Thailand
E-mail: sonnt3377@yahoo.com
Problem identification: Given a multicast tree with source node S and set of destination node Γ, our work is to find paths from S to all \( T \in \Gamma \) so that those paths satisfy certain conditions. The set of conditions may contain only one or more than one condition (multi-constrained multicast routing). The parameters for routing may be bandwidth, delay, delay jitter, or their combinations.

### 2.2 Multicast Distributed Routing Algorithm (MDRA)

Similar to [2], our algorithm uses three kinds of message: probe, acknowledgement (ack) and failure messages. The path determination is done by the following two phases: probing phase and ack phase.

In probing phase, each node is waiting for a probe message to arrive. Having a probe arrival, the node checks (based on one information in the probe) whether it is in the tree.

If it is not a tree node, the node, again, checks whether this is the first probe. If this condition is true, then node records the predecessor node and forwards probe to the outgoing interfaces satisfying forwarding condition of MDRA. Otherwise, it simply discards the probe.

If this is a node in the multicast tree, this node also checks whether this is the first probe. If so, node sends ack message to the predecessor (in order to reserve resources), records the predecessor node and forwards probe to the outgoing interfaces satisfying forwarding condition of MDRA. Otherwise, it simply discards the probe.

The forwarding condition at each node is proposed in a generic form so that more customization could be done later.

\[
\text{for (all outgoing interfaces except the one probe comes) if (QoS condition is satisfied) then forward packet}
\]

We would like to find the paths satisfying the following conditions: bandwidth is greater than or equal to B and delay along the path must be less than or equal to D. The forwarding condition is:

\[
\text{for (all outgoing interfaces except the one probe comes) if ((link-bandwidth \geq B) \& \& (link-delay + cumulative-delay) \leq D) forward packet;}
\]

By using selective probing mechanism [2], we can save network resources, since the number of forwarding packets is limited. However, the trade-off is that, the successful ratio decreases. We will use simulation to verify the acceptable range of this value.

Figure 1 shows the procedure for processing probe message, while Figure 2 shows the procedure to work with ack message.

![Probe message processing](image1)

![ACK message processing](image2)

The acknowledgement phase starts when an ack arrives at a node. The processing procedure for ack message is quite complicated, since the algorithm works with more than one source and destination, and with different QoS requirements (of different type of connection).

Having the ack message arrives, the node checks whether it has enough resources required by the ack. If not, it sends a failure back to the successor (the node that sent ack) to refuse the connection. If the node has enough resources, it then reserves resources, records the successor (the node that sent ack), and continues checking procedure.

If this node is source node, then connection is established.

If not, the node checks whether this is the first ack. If so, it sends ack to predecessor, which is already recorded when the probe message came. If this is not the first ack, node checks if a new ack requests more resources then allocates more if possible (and sends new ack to upstream node to request more resources). If the reserved resources satisfy new ack requirement, then the node discards this ack.
Figure 3 represents processing procedure for failure message.

2.3 Complexity Analysis

The algorithm takes a single message round-trip time to establish a connection. Assuming that in normal condition, it takes at most one unit of time for a message to traverse one link including the buffering and processing time at nodes, then the time complexity is $O(2l)$ units of time, where $l$ is the length of the tentative path.

For large network, the number of edges $E$ is normally much greater than the length of the tentative path; hence, time complexity of DRA is bound by $E$, or $O(E)$.

For MDRA, with the number of destination is $M$, then the time complexity of MDRA is $O(M.E)$. This value is smaller than the ones of [6] and [9], which are $O(V^3)$ and $O(kiMV^4)$ respectively. Results in [5] are better than MDRA, which is $O(2.M)$, but this one is only used for optimal cost, delay-bounded parameters and could not use as a generic framework for other parameters. More comparison about the complexity of these algorithms can be seen in [10].

The algorithm sends at most one probe per link in the subnet consisting of all paths from the source to destination. The total number of probes sending is bounded by $E$. There are at most one ack and one failure message for each link on the tentative path. The total number of ack and failure message is bounded by $2l$. Hence, the message complexity (number of control message) of the algorithm is $O(E+2IM)$ for a single connection request (the overhead of failure and retry is not included). For very large networks, $E$ is much greater than $2IM$, hence, the message complexity is $O(E)$. Again, the message complexity of MDRA is less than the one in [6], which is $O(V^3)$. [9] and [5] provide better results in term of message complexity (consider zero and $O(2.M)$ respectively). However, [9] must hold a global state at each node while [5] is applied for delay-bounded parameter only.

Note: $k$ and $l$ are constants in the algorithm. A larger $k$ or $l$ results in a higher probability of finding a feasible tree and higher overhead.

3. Simulation Results

3.1 Simulation Procedure

We use simulation to evaluate the performance of MDRA. Our simulation has been carried out by putting delay into account as a constraint parameter. The simulation with other parameter and/or combination of them should be treated similarly. For NP-complete cases, there should be heuristic algorithm to solve the problems.

For the purpose of simplicity, we carry out the simulation with one source and a number of destinations.

Two topologies are used in the simulation. The simulation methods for two topologies are the same. However, the effects are different for small and large networks.

The inputs of our simulation contain the following components: a specific graph $G = (V, E)$, which is chosen at the initialization of the simulation work. This graph is created manually and stays unchanged during the whole simulation; a source node $S$ and a set of destinations $\Gamma$, which are generated uniformly and randomly at the beginning of each iteration. The number of nodes in $\Gamma$ is greater than or equal to two at each iteration, $S$ and $\Gamma$ are regenerated. The simulation results are averaged based on the number of iteration. The more iteration, the more accurate results.

QoS requirement is delay of connection. This value is ranging from 100 to 500 ms with QoS routing. For non-QoS routing, there is no delay constraint (or the delay constraint is infinity). For each link in the topology, the link delay is randomly generated, and uniformly distributed in [0, 125] ms. Thus, the average link delay is 62.5 ms. Notes that, link delay mentioned here is the combination of queuing delay, processing delay, transmission and propagation delay.

As mentioned above, we evaluate the effectiveness of our
algorithm in term of delay parameters. The comparison is made based on performance of the followings three mechanisms:

- **non-QoS routing mechanism**: this is a normal routing mechanism based on given metrics (delay, bandwidth or any other parameters). There is no constraint at all.
- **QoS Multicast Routing with flooding mechanism** (called QoS flooding hereafter): this mechanism is QoS routing, since packets are forwarded to all interfaces satisfying forwarding conditions.
- **QoS Multicast Routing with selective probing** (called selective probing hereafter): selective probing is an extension of flooding mechanism when packets are forwarded to some interfaces satisfying forwarding condition.

In this work, the performance has been evaluated based on the following criteria:

- **Routing overhead**: the number of packets sent to carry path determination. They should be less than the one of flooding mechanism (broadcast).
- **Successful ratio**: this parameter is less than or equal to the one of flooding mechanism. However, we hope that this ratio is still in an acceptable range. Successful ratio is defined as follows:

\[
\text{Successful ratio} = \frac{\text{number of accepted connections}}{\text{number of requested connections}}
\]

Where number of connection requests is the number of request generated by simulation program. The larger this number, the more accurate results.

### 3.2 Results

The following simulation results are averaged over one hundred iterations.

**Figure 6**: Number of packets per connection request (topology-1)

As shown in Figure 6, the average number of packets (probe messages) needed for a connection request is 17.56 for non-QoS routing. This one decreases when QoS routing is applied. For QoS routing by using flooding mechanism (to broadcast packets to all outgoing interfaces satisfying forwarding condition), the number of packets needed for a connection request reaches the one of non-QoS case quite fast as the delay increases (hence, more outgoing interfaces satisfies the forwarding condition). For delay constraint of 100 ms, the packets for flooding QoS is 12.09 per connection request and reach to 17.18 when delay constraint is 200 ms.

In case of selective probing, the number of packets needed for each connection request is smaller, due to the limited packet forwarding. For delay constraint of 100 ms, this value is 8.14, and reaches to 11.70 when delay constraint is 500 ms. Those values are smaller than the one of non-QoS and QoS flooding, which are 17.56, 12.09, 17.56 and 17.56 respectively. By using selective probing mechanism, we can significantly reduce the number of packets needed for path determination phase (reduce network overhead). That is, we can save more bandwidth.

**Figure 7**: Call admission ratio (topology-1)

Figure 7 represents the successful ratio in three cases: non-QoS (using normal link state routing protocol), QoS routing (using broadcast forwarding mechanism) and QoS selective probing. Y-axis is the successful ratio of the connection request while X-axis is the delay as shown in Figure 6.

According to the above analysis, selective probing can reduce network overhead (as shown in Figure 6), but it reduces the successful ratio of the connection as well, due to the use of less number of packets for path determination. This section shall provide some simulation results to evaluate the trade off between network overhead and successful ratio.

The successful ratio of non-QoS case is considered 100%, since network can always find the path to a specific destination if network failure does not happen (all the links and nodes are always working and there is no dead-lock at any node). The successful ratio of the QoS flooding mechanism is less than non-QoS which depends on the delay constraint of the connection. If the delay constraint is small (100 ms), it is very hard to find a path satisfying the delay constraint (since the average delay of each link is 62.5 ms – as pointed out previously). That leads to low successful ratio of 79.45% comparing to 100% of non-QoS. However, when the delay constraint is large enough, comparing to link delay, the successful ratio reaches the one of non-QoS (95.82% with 150 ms and 99.06% with 200 ms delay).

In selective probing case, the successful ratio is smaller due to the limited number of forwarding packets (the number of “test-paths”). That value is small (66.55%) when the delay is
small (100 ms), comparing to link delay, but can be quickly improved as the delay increases (96.29% when the delay is 250ms). Note that the average link delay is 62.5 ms.

Figure 8: Number of packets per connection request (topology-2)

Figure 9: Call admission ratio (topology-2)

Figure 8 and 9 represent the same evaluation for delay parameter. However, the results are taken from simulation for topology-2, which is more complicated (more nodes and links). The average number of packets needed for routing is, therefore, higher since there are more chances that nodes are far away from each other. The successful ratio is also smaller, comparing to the case of topology-1. For small delay (100 ms), it is nearly impossible to establish connection between two nodes (since the average node distance is more than 2 links). That results in a very small successful ratio for small delay. Those values are 5.93% for QoS-flooding mechanism and 2.62% for QoS selective probing.

With a reasonable delay for the connection (500 ms), the successful ratio of MDRA (delay) is acceptable value (92.09% for selective probing and 99.84% for QoS flooding). This is quite a good result if we take the ratio of delay (500 ms) and average link delay (62.5 ms or the distance between two nodes is 12,625 km using copper cable, which has propagation speed of two third of the light speed) into consideration.

The number of probes per connection request for selective probing is significantly less than the one of QoS flooding and non-QoS case (which is 49.35, 69.17 and 69.33 respectively – for delay of 500 ms). Thus, selective probing can save more resources as the network size increases.

4. Conclusions

In this work, we proposed a distributed algorithm for Multicast Routing. Each node is required to maintain only its local state, hence, reduce the requirement of exchanging message between node and save bandwidth.

The MDRA algorithm is evaluated by analysis as well as simulation. Performance of the new algorithm is investigated and compared to the one of QoS routing using flooding mechanism and broadcast routing (non-QoS). It is found that MDRA reduces the network overhead significantly (about 33 – 35% in our simulation) while still keeping the Call admission ratio in an acceptable range (greater than or equal to 90%). Thus, the performance of MDRA is better than the one of flooding mechanism.

The results and analysis show that, using selective probing mechanism can provide better result (in term of routing overhead) while still keeping the performance (successful ratio) in a reasonable range.

References


