Propose a new algorithm of routing with information collected locally

Tran Minh Anh and Nguyen Chien Trinh

Abstract—The type of algorithm which uses local information collected from source node for Quality of Service (QoS) routing has recently been researched as an alternative to QoS routing algorithms that traditionally use global state information. This algorithm, collecting information from source node only, helps flow routing better and assures more flexibly QoS for network. This trend leads to a new solution for satisfying the higher and higher demand of telecom market in the near future. In this paper, we introduce a new algorithm of routing like that type for assuring the quality of network as well as quality of services. The simulations at last section show the advantages over some other localized routing algorithms and global routing algorithm.

Index Terms—QoS routing; algorithm; localized; information.

1. Introduction

In the coming time, using telecom network with very high speed and high quality becomes quite popular; hence the assuring of quality of network becomes much challenged. The way of using local information at source node about congestion probability, network statistics . . . to build sets of paths available for routing is presently an effective method to convey information in network. When local information is used, this routing technique will decrease the average overall blocking probability of packets transmitted through the network as well as improve the overall performance of network. It has been demonstrated that this technique is simpler and better than traditional global QoS routing schemes which commit several problems, such as: a large communication overhead, the inexact of global state and the out-of-date information due to large update intervals. Differently, the type of localized routing tends to solve these problems by routing with information collected locally. It routes flows using this localized view of the network QoS state through the probability of blocking, flow-through . . .

In this approach, each node builds, maintains a predetermined set of candidate paths to each possible destination and routes flows along these paths. Therefore, the localized method will open a new approach to assure QoS for network in the near future.

In this paper, we propose a new algorithm of routing (we call rbda: QoS Routing using Bandwidth-Delay based Algorithm), a localized QoS routing algorithm in which the two parameters of QoS (bandwidth and delay) are used as criteria for routing, accompanied with an index of counting number of successfully transmitted flows. We compare and realize better performances against other localized schemes and the traditional global QoS routing algorithm Widest Shortest Path (wsp) when performing them with the same types of topology, traffic patterns and range of traffic loads.

2. Related works

Recently, the researches in [1-9] on QoS routing which have been done much, and from that we can see that the parameters of QoS like: Bandwidth, Delay, Jitter, Packet Loss . . . are now more and more important for telecom applications in the future telecom network. To meet the demand for QoS parameters, QoS Routing is an effective method which is used much in telecom network. It helps to find a feasible path to satisfy the flow requirements. To do this, it demands the knowledge of the network state and this knowledge should be kept up-to-date and as accurate as possible. The routing protocols help to make this information available for nodes when making routing decisions.

Any routing strategy consists of two basic tasks. The first task is to collect the link-state information and keep it up-to-date. The second task is to compute the most feasible path for each flow based on the collected state information. In global QoS routing algorithms, nodes need to exchange network state information to get a global view of the network state. Based on that knowledge and the current state information, a node can compute a feasible path for the current flow towards the intended destination.

Many currently deployed algorithms follow this approach, and they implement different methods to compute feasible paths. Global routing algorithms usually make a trade-off between the network state update and the accuracy of state information. They also tend to achieve optimal performance by balancing the traffic...
load across the network and efficiently utilizing available resources. One of the most popular global QoS routing algorithms is the Widest Shortest Path algorithm (WSP) [2]. WSP was proposed to balance the traffic across the network by finding a feasible path with minimum hop (minhop) count. If more than one path with the same hop count is found, the one with the largest residual bandwidth is selected. WSP is a link-constrained path-optimization routing algorithm as it finds the shortest feasible path. The widest path criterion is used only when there are several paths with the same hop count to choose from.

WSP minimizes the usage of resources by preferring shorter paths over longer ones, which leaves more resources to handle other flows. WSP will be used to compare without proposed algorithm in section 4.

As mentioned in the first section, the traditional global QoS routing algorithm commits a lot of problems due to the large communication overhead, the inexact of global state and the out-of-date information due to large update intervals.

After some recent researches, localized QoS routing algorithm is a new approach in the telecom networks based on the idea of using local information to route packets through the network. One of these schemes is the scheme of the localized Credit Based Routing algorithm (cbr); see [10], which will be used to compare with the performance of our algorithm.

As well as other localized routing algorithms, the cbr predetermines a set of candidate paths P between each pair of source and destination from minhop set R_{min} and alternative set R_{alt}. In the algorithm, the cbr builds a crediting scheme to reward and penalize a path based on the statistics about flow routing in the network.

The cbr updates frequently the credit for all paths after its times of transmitting flow. If a path transmits successfully a flow, its credit will increases and vice versa. The value of credit finally reflects the flow blocking probability of that path, and this mere value is used to select path itself for next incoming flow. The larger the credit of path is, the more chances for that path to be selected are. It means that the cbr selects the path with largest credit in each set R_{min} and R_{alt} upon flow arrival. Then, the cbr uses the system parameter \( \Phi(\Phi \leq 1) \) to decide the route for flow with the formula as below:

If \( R_{\text{min,credits}} \geq \Phi x R_{\text{alt,credits}} \), the flow is routed along the minimum hop path \( R_{\text{min}} \).

And if \( R_{\text{min,credits}} < \Phi x R_{\text{alt,credits}} \), the flow is routed along an alternative path \( R_{\text{alt}} \).

Based on the statistics of the path blocking probability, the cbr increases or decreases the path credits upon flow acceptance or rejection. Besides, with that blocking probability, the cbr fixes a MAX_CREDITS parameter to determine the maximum attainable credits for each path, as follows:

\[ 0 \leq R_{\text{credits}} \leq \text{MAX_CREDITS}. \]

To do that, the cbr algorithm uses a sliding window for \( M \) connection requests, and records rejection and acceptance for each path. It uses 1 for flow acceptance and 0 for flow rejection in that window. With the value taken from this window, the cbr will calculate each path blocking probability for the period of \( M \) connection requests. The main problem with cbr is that a path’s credits are only updated each time that path is selected. If a path is selected infrequently, its credit value will become stale leading to errors in the selection process.

Another scheme of QoS Routing using local information is the Proportional Sticky Routing (psr) algorithm proposed in [11-14] which will also be used to compare the performance of our algorithm.

The psr has main idea is that: Every source node predetermines and maintains the set of candidate paths \( P \) created by the set of minhop paths \( P_{\text{min}} \), and the set of alternative paths \( P_{\text{alt}} \). Then, based on statistics of the number of blocked flows and their flow blocking probability at itself, a source node distributes proportionally the traffic load to a destination among that predetermined set of candidate paths \( P \).

In operation, the psr algorithm works in observation periods with varied-length cycles. In each period, based on the flow proportion and the flow blocking probability information, source node selects path for routing flows. If the path is selected more times, it will get bigger proportion which affects the next selection for flow-in. At source node, the set of eligible paths \( P_{\text{els}} \) is set at first from the set of candidate paths \( P \). For each cycle, the flow-in can be routed among paths from the set \( P_{\text{els}} \).

A parameter \( \gamma_r \), called flow blocking parameter is set to determine the times of blocking a connection request of a path. If a path has the times of blocking in excess of \( \gamma_r \), that path becomes ineligible. When all the paths in the set \( P_{\text{els}} \) become ineligible, the cycle will end, and the next cycle begins. The psr select a path based on its flow proportions as mentioned above. The more times the path is chosen, the more chances for selection.

After each observation period, the minhop path flow proportions are adjusted to equalize the blocking probability among paths. For the alternative paths with minhop+1, the minimum blocking probability among the minimum hop paths is also used to control their flow proportion. After that, another period begins.

With the proportional distribution like that, the psr helps the network more balancing, and utilizes the network more efficient.

Using end-to-end Delay to be constraint, many other localized routing algorithms were proposed recently such as [15]. One of them is Localized Delay-Constrained QoS Routing algorithm (ldcr) [15].

As well as other localized routing algorithms, ldcr needs all source nodes to predetermine and maintain the set of candidate paths. Being a source routing algorithm, ldcr requires the source node select in the candidate path set the path for flows. The path which has the best quality is selected by ldcr to route flow if it satisfies the flow QoS requirements.

To specify the best path, ldcr uses the value of average end-to-end path delay of that path to decide whether the path is good or not for routing. ldcr collects
all information at source node itself; the information is blocking probabilities, flow blocking ... Besides, the value of end-to-end delay of each candidate path is the main metric for comparison of selecting path. From that, the path with the least value of average end-to-end delay is selected to route that flow. And also, that average end-to-end path delay is a constraint for estimating path quality. After a flow is routed, ldr can update the average delay of that path, and the loop continues with this value of delay.

With the mechanism like that, ldr can assure the end-to-end QoS delay for users and optimize the performance of the entire network.

The schemes above will be used to compare with our proposed algorithm through simulations. The results of simulations will be showed in the section 4.

3. Proposition QoS routing algorithm with information collected locally

3.1. Methodology

Nowadays, when telecom services unify in the same background of network, these services, especially interactive services, demand high quality of network such as large bandwidth, shorter delay time ... Therefore, network providers must assure the quality of network for customers who use these high quality services, and it should be able to support the increase demands by applications to provide different classes of services to suit the diverse QoS requirements. These service requirements impose strict constraints on transporting networks to ensure end-to-end performance guarantee.

The QoS requirements are in a set of constraints for which routing algorithm must satisfy to find out the feasible paths for flows. The QoS parameter specifies type of network quality, i.e.: Bandwidth, Delay ... If there is not services’ QoS requirement, there is not corresponding constraint for routing.

To analyze the characteristics of services about QoS requirements, we see the tables below.

### TABLE 1

**TYPE OF SERVICES’ CHARACTERISTICS**

<table>
<thead>
<tr>
<th>Services</th>
<th>Characteristics</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>Non real-time</td>
<td>- Data/Internet access</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- FTP download/upload ...</td>
</tr>
<tr>
<td>Real-time</td>
<td>Real-time broadcast that cannot be</td>
<td>- VoIP, IPTV, VoD</td>
</tr>
<tr>
<td></td>
<td>recreated once lost</td>
<td>- Internet radio, TV, Internet gaming, Video</td>
</tr>
<tr>
<td></td>
<td></td>
<td>conference ...</td>
</tr>
<tr>
<td>High-priority</td>
<td>Mandatory traffic to maintain</td>
<td>- OAM frames, Switching or routing control frames</td>
</tr>
<tr>
<td></td>
<td>stability in the network</td>
<td>- Network synchronization such as SynchE, 1588v2</td>
</tr>
</tbody>
</table>

With the types of services’ QoS requirements above, ITU-T in the recommendation Y.154x in [16] has proposed the requirements for QoS standard as Table II.

<table>
<thead>
<tr>
<th>Class</th>
<th>Application/Examples</th>
<th>Mean Delay upper bound</th>
<th>Delay variance</th>
<th>Loss Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 0</td>
<td>Real-time, jitter-sensitive, high interaction (VoIP, Video Teleconference)</td>
<td>100ms</td>
<td>&lt;50ms</td>
<td>≤1.00E-3</td>
</tr>
<tr>
<td>Class 1</td>
<td>Real-time, jitter-sensitive, interactive (VoIP, Video Teleconference)</td>
<td>400ms</td>
<td>&lt;50ms</td>
<td>≤1.00E-3</td>
</tr>
<tr>
<td>Class 2</td>
<td>Highly interactive, transaction data (e.g signaling)</td>
<td>100ms</td>
<td>Uns-specified</td>
<td>≤1.00E-3</td>
</tr>
<tr>
<td>Class 3</td>
<td>Interactive transaction data</td>
<td>400ms</td>
<td>Uns-specified</td>
<td>≤1.00E-3</td>
</tr>
<tr>
<td>Class 4</td>
<td>Low loss only (Short transactions, bulk data, video streaming)</td>
<td>1s</td>
<td>Uns-specified</td>
<td>≤1.00E-3</td>
</tr>
<tr>
<td>Class 5</td>
<td>Default IP networks applications</td>
<td>Unspecified</td>
<td>Uns-specified</td>
<td>Uns-specified</td>
</tr>
</tbody>
</table>

In the table, we can see that IP networks are now carrying a heterogeneous mix of traffic, with widely differing QoS requirements. The service model of emerging multi-service packet networks is based on the network’s ability to guarantee QoS to user applications.

Because the current network is very large and highly hierarchical, QoS routing becomes very important to select paths which satisfy QoS requirements. The path selection is far much complicated and challenging, due to the fact that multiple QoS constraints are required to be met, and at the same time keeping track of the ongoing dynamic changes in the network resources and topology. It is the responsibility of routing algorithms to tackle these problems while achieving high routing performance. Besides, in the QoS metrics, bandwidth and delay time are currently the most significant QoS performance metrics for customers’ services. Although using many metrics to be criteria will make computation becomes too much, it will help routing more flexibly and more sophisticatedly. Therefore, we take these two metrics Bandwidth and Delay as criteria in our scheme, and we call our scheme as QoS Routing using Bandwidth-Delay based Algorithm or rbda.

3.2. Describing the algorithm of rbda:

3.2.1. Introduction:

As proposed, our algorithm (rbda) also requires every node to maintain a predetermined set of candidate paths R to each possible destination, and to keep track for only one set of candidate paths R (the result of finding shortest paths).

Unlike the previous localized QoS routing schemes like cbr which uses CREDIT to reward or to punish path, or psr in which the number of blocked flows was only available information at the source node, or ldr, which uses average end-to-end path delay to make routing decision and updates the selected path’s average delay to be quality of the path, our scheme is quite different. Here we use the values of bandwidth and delay from local node to be criteria for choosing path. This scheme becomes more flexible than cbr, psr or ldr which will be used to compare in the simulation with our algorithm.
3.2.2. The algorithm of rbda:

In our scheme, every path \( P_i \) in \( R \) associated with a variable \( P_i, \text{Quality} \in [1,2] = P_i, \{\text{Bandwidth}, \text{Delay}\} \) with the count index of that path, called \( C_i \) and total flow accepted on that path between source and destination called \( N_i \). The way to build \( C_i \) will be discussed later. And, to choose path, the set \( R \) (all of candidate paths) will be ranged for \( C_i, N_i \) to choose the \( \{\text{max}(C_i), \text{min}(N_i)\} \), we call that the set \( R \) is ranged for \( \max \{C_i, N_i\} \).

The process of routing:

When flow arrives, \( \text{rbda} \) will select the path \( P_i \) with \( \max \{C_i, N_i\} \) (first place in set \( R \) after ranging). It first of all checks the demand of the flow and uses it for choosing path, we call that is \( \text{RQ}, \{\text{Bandwidth}, \text{Delay}\} \) or \( \text{RQ} \) (Requested Quality). If there is not any QoS requirement from flow, it will choose the path \( P_i \) to route. Otherwise, it compares \( P_i, \text{Quality} \) and \( \text{RQ} \) (the demand). It normally takes the value of \( \text{RQ} \) from Service-Level Agreement of flow. From the comparison between \( P_i, \text{Quality} \) and \( \text{RQ} \), we have two cases:

1) \( P_i, \text{Quality} \geq \text{RQ} \): the \( P_i \) will be chosen.
2) \( P_i, \text{Quality} < \text{RQ} \).

(The way to compare will be discussed in part of 3.4).

\( \text{rbda} \) do the loop until finding out the path has maximum of \( C_i \) (with minimum of \( N_i \)) and \( P_i, \text{Quality} \geq \text{RQ} \). If no path has quality satisfying \( \text{RQ} \), the arriving flow obviously is denied. \( \text{rbda} \) increase \( C_i, N_i \) if route \( P \) is accepted, decrease \( C_i \), keep \( N_i \) if route \( P \) is unaccepted accordingly. All \( C_i \) will be decreased if no path is chosen (it means no path has quality better than \( \text{RQ} \)).

The counting of \( C_i, N_i \) values will be discussed clearly in the next part.

3.2.3. Flow chart of all stages:

With the algorithm described above, we can observe the algorithm operates as the flowchart below:

![Flow chart of rbda](image)

**Fig. 1. Flow chart of rbda**

\[ N_i = (N_i + 1) \]  \hspace{1cm} (2)

and when the path is not chosen (for its quality is minor than \( \text{RQ} \)), or unaccepted:

\[ C_i = (\text{Last } C_i - 1) \]  \hspace{1cm} (3)

\( C_i \) is not below 1, if \( C_i \) is below 1, \( C_i \) gains 1

where \( T \): a parameter is set first, depends on the capacity of network. Last \( C_i \): the value of \( C_i \) before next flow comes to source node, \( N_i \) is number of flows accepted by that path.

After that, the following flow will use this \( \{C_i, N_i\} \) to be criteria to choose path, and next loop re-begins. Therefore, after one flow processed, \( \{C_i, N_i\} \) changes accordingly to the success/fail ratio of that path, and probability of that path changed correspondently. It affects to the probability of being chosen for the next.

The value \( T \) depends on capacity of network. It means after maximum of \( T-1 \) successive time of “fail” comparing, the value \( C_i \) will gains value 1, it will leads to rebuild set. So, to assure the quality of transmitting flows, we choose \( T \) about 10-20, according to network of 32 nodes in simulation.

After one set of candidate paths between a pair of particular source-destination has \( C_i = 1 \) for all paths, the source node should collect information from global network to rebuild. The renewal of set will help algorithm to work better with new information from network. After each time of rebuild, all the \( C_i \) will be reset at \( T \) and all the \( N_i \) will be 0.
After half of sets have been rebuilt, the source node should rebuild all of sets. The rebuilding of sets of paths will refresh all routing information, add some more new “join-in” nodes and discard some “disjoint-out” nodes from the collected local information.

3.3. The pseudo-code of algorithm

```
Initialize
Building R
Set T=10
Set Ci=T, ∀ Pi∈R
Set Ni=0,
rbda
1. Range max{Ci,Ni} for flow-in
2. Set P_i.success = false;
3. Set RQ for flow-in
4. Pi=first(P, (Ci,Ni): Pi∈R)
5. While !(P_i.success or R(end))
6. if(P_i.Quality≥RQ)
7. Route flow along path P_i
8. if P_i is accepted
9. {Compute P_i, (Ci,Ni) (increase C(≤T), Ni)}
10. P_i sucessoe=true)
11. else
12. elseif
13. P_i= next(P_i, (Ci,Ni): Pi∈R)
14. Compute P_i, Ci (decrease C(≥1))
15. endif
16. ElseIf
17. P_i= next(P_i, (Ci,Ni): Pi∈R)
18. Compute P_i, Ci (decrease C(≥1))
19. EndIf
20. EndWhile
21. END.
```

Fig. 2. The rbda pseudo code

3.4. The metric selection and comparison

As mentioned before, we choose Bandwidth and Delay as the two main metrics for comparison to choose paths. There are some ways to compare, but the two relevant and popular ways will be discussed below:

3.4.1. The single mixed metric

For simplicity of computation, we propose the only metric, that is called mixed metric, and signed q(B_i/D_i) as follows:

\[ q(B_i/D_i) = B_i/D_i \]

where B_i: the minimum residual bandwidth of any link on path and D_i: the sum of all delay of propagation from all links on this path.

We call \( f(B_f/D_f) = B_f/D_f \) as the value of that flow. The comparison for choosing path:

\[ q(B_i/D_i) \geq f(B_f/D_f) \]

If the result is true, it means this path has enough quality for that flow and if it is false, the path is ejected and we choose another path.

3.4.2. The comparison for the two metrics

We can make the comparison of the two metrics independently. First, we compare the bandwidth of the path with the demanded bandwidth of flow-in. If it’s true, we continue to compare the Delay of that path with the demanded delay for that flow.

The path is only chosen when we have two values of true for two comparisons.

(Note: We compare \( P_i, Bandwidth \geq F, Bandwidth \) and \( P_i, Delay \leq F, Delay \), where Bandwidth and Delay of path are determined like above (in 3.4.1).

The procedure is as follows:

```
void Routing2::compare(int pathB, double pathB, int flowB, double flowD)
{
if (pathB == flowB)
if (pathD <= flowD)
    chosenpath(P_i);
else
discard(P_i);
else
discard(P_i);
}
```

Fig. 3. Using comparison with two constraints Bandwidth and Delay

In the experiments, we interchangeably use both ways above and the results get almost the same. But, the results displayed in the paper are from the latter one.

3.4.3. Other ways to compare

There are also other ways to compare. For example:

- Estimating and using probability of some metrics
- Quantizing values of metrics
- Segmenting the scope or range of metrics …

Some other methods often use large memory in overhead to do, so it’s rarely applied in reality. Hence, in the scope of this paper, we don’t mention much about this, although some ways are still effective to use in routing with some criteria.

4. Performance evaluation

In this section, we realize the performance of the rbda scheme and compare it with the cbr, psr and lcr schemes. Besides, we also compare with the global QoS routing scheme widest shortest path (wsp). All the experiments will be set in the same condition. Next, we analyze the results of our simulation model and performance metrics of all the schemes.

4.1. Simulation Model

We use simulator based on OMNeT++ [20], an event-driven simulator which is used commonly now. To evaluate the results, we collect all of parameters of simulation as vectors, scalars and histograms to compare. The setup of simulation experiments is similar to the simulation in [10-15], described as follows: network built with 32 nodes, links of these nodes are all bidirectional with the same capacity \( C = 150\text{Mbps} \) in each direction and
the same value of delay $D = 10ms$, flows arrive to each source node according to a Poisson process with rate $\lambda$ and destination nodes are randomly selected (each node is capable of being both source and destination), flow duration is exponentially distributed with mean $1/\mu$, flow bandwidths are uniformly distributed within [0.5-4MBytes]. Network is performed in Fig. 4.

We also calculate the overall end-to-end delay of network when we use the scheme rbda in comparison with wsp with small load and high load. From those results, we can conclude the effectiveness of the rbda scheme.

4.2. Simulation Results

From the simulation results of flow blocking probability, we collect information and compare with the ones of other schemes, as shown in Fig. 5 to Fig. 8.

From [17-18], the offered network load is $\rho = \lambda Nbh/\mu LC$ where $N$ is the number of nodes, $b$ is the average bandwidth required by a flow, $h$ is the average path length (in number of hops) and $L$ is the number of links in the network.

In the experiments, we set $N=32$, $L=108$, $h=3,137$, $1/\mu = 60s$. Since the performance of routing algorithms may vary across different load conditions, our simulation experiments consider several types of different load conditions through the value of $\lambda$ according to experiments of from low loads to high loads.

To compare with other schemes, we choose flow blocking and bandwidth blocking probabilities as criteria as well as the simulation in [10-15]. The blocking probabilities are calculated based on the most recent 100,000 flows. The time interval of simulation is set about of 20 minutes, equivalent of more than 2.5 millions of flows emitted. Then, the standard overall flow blocking probability is defined as:

$$\text{Flow Blocking Probability} = \frac{|B|}{|T|} \quad (6)$$

where $|T|$ is the total of all flows and $|B|$ is total of blocked flows. Besides, we calculate bandwidth blocking probability which is defined as:

$$\text{Bandwidth Blocking Probability} = \frac{\sum \text{bandwidth of } |B|}{\sum \text{bandwidth of } |T|} \quad (7)$$

Fig. 5 and Fig. 6 show the performance of rbda against cbr, psr, ldcr and wsp in terms of flow and bandwidth blocking probabilities under load $\rho$ varies between 0.2 and 0.5. From these values, we can see that under low load ($\rho \leq 0.25$), the difference in the performance of the routing algorithms is quite small, because finding available path with sufficient bandwidth is easy; and flows are almost accepted.
bandwidth blocking probability is high at the same time as view in Fig. 5 and Fig. 6.

The reason of these differences: In the case of the *rbda* scheme, based on probability of being chosen, when paths are selected, the index of that path increases. This assures that this path is good and may support relatively for the next flow. It means that it has already available quality in the links on the selected path. Other than *cbr* or *psr*, *rbda* uses both Bandwidth and Delay as criteria to compare, so all flows which are compared, almost satisfy the demand of Delay at destination, so it helps decreasing flow blocking probability of that path, and also decreasing the value of End-to-End Delay of those flows.

Moreover, the setting index for choosing path helps to avoid the congestion of flows come at nodes simultaneously, particularly, when the load increases and the links begins to become congested. If congestion happens, flows will be re-directed to other path and the index decreases at once. Then, the source node might diminish the using of these paths which have low index. Therefore, the probability of flow blocking and bandwidth blocking is considerably low against the case of *cbr*, *psr*, *ldcr* and *wsp* as well.

In the Fig. 7, we collect information about metrics of Average End-to-End Delay in the different load (with load $\rho$ up to 0.8). The required value set for Delay of each packet is randomly distributed between 20ms and 250ms.

We calculate the results of our case *rbda* with the case of *wsp* (using Dijkstra algorithm [19] with weighted links to route). It shows that when the load augments, the more flows are through network, *rbda* expresses the more efficiently. The End-to-End Delay of *rbda* is quite the same as *wsp* at the small value of $\rho$, but, when the load $\rho$ increases, it becomes better as showed in Fig. 7.

In Fig. 8, we fix the load $\rho$ at 0.8, then, we calculate parameter End-to-End Delay in this case. From the Fig. 8, we can see that when the number of flows increases, the Average End-to-End Delay will keep a stable value as showed in the Fig. 8.

In our case *rbda*, the Average End-to-End Delay is higher but keeps in a margin of 38.3-38.4ms. But, in the case of *wsp*, that gains higher value, within about 39.7-39.9ms. It means that with high load, the congestion happens more frequently, so, this value increase dramatically, especially in the *wsp* case (see Fig. 8). In our case, the flows in our cases must change path frequently based on the index $C_i$. When congestion happens, the index of the regular path diminishes, then, our case changes path. Then, the average End-to-End Delay is better than case of *wsp* as well.

In concluding, the case of *rbda* has better performance than other cases such as *cbr*, *psr*, *ldcr* or *wsp* in some experiments which have been done.

### 4.3. Complexity and overhead

The case of *wsp* uses the algorithm Dijkstra, like almost global QoS routing algorithms, takes at least $O(N\log N+E)$ time, where $N$ is the size of the network measured in the number of nodes, and $E$ is the number of links (edges), see [2]. At the same time, the localized schemes use the way of routing that selects path from the set of candidate paths $R$, with the size of $R$ is $|R|$.

In the *cbr*, *psr* and *ldcr* algorithms, the path selection is an invocation of a weighted-round-robin like path selector, whose worst case time complexity is $O(|R|)$, similarly *rbda* requires order of better than that, for it uses only the minhop of paths as explained above. In addition these localized schemes require updating information, which takes a constant time $O(1)$ as [10]. Therefore, with communication overhead, *rbda* or other localized schemes require very little over and above computing the blocking probability based on acceptance or rejection of a path, while at the same time, global algorithms require a huge amount of overhead to keep the link state information updated. In conclusion, the computation of our scheme at source node anyway is much smaller than the one of traditional *wsp* cases.
5. Conclusion and ongoing work

In this paper, we propose a new localized QoS routing algorithm to choose path using only flow information collected locally at source node. We have done many experiments to compare the performance among rbda, cbr, psr, ldcr and wsp algorithm; and show a considerable performance with better time complexity and very low communication overhead.

The flow chart and pseudo code of the algorithm have been proposed with the two metrics bandwidth and delay, QoS metrics, to route flows through network. Using these tools, the local information of blocking probability could be used to update the path quality through our proposed index. This index directly decides the routing, hence makes better quality of routing, on the other hand, better working of network.

As part of future work, we will investigate the ability of this algorithm in using more QoS parameters to compare, loss packet or delay jitter and so on. It will of course make a lot of computation, and be very complicated, but in the very high speed network, it will become very important, when all services will use the same infrastructure, and the telecom services will surely require very large bandwidth.

Another part of future work of this algorithm is to see the balancing of the network in building routing policy. The algorithm should care for it, and the factor of balancing should be a new parameter in routing.

And finally, as proposed in the paper, the selection of set of candidate paths plays very important roles. In the future, with the more flexible selection of set of paths, the algorithm will operate more effectively and more reliably.

References


Tran Minh Anh received his BSc in Electronics and Telecommunication Engineering from Danang University of Technology (DUT), Danang, Vietnam, in 1995. In 2001, he received his MSc in Telecommunication Engineering from Hanoi University of Science and Technology. Before starting his graduate studies, he worked for the Vietnam Posts and Telecommunications Group (VNPT) as an engineer. He was involved in a variety of work concerning his profession as a telecom engineer at VNPT from 1995. His research field concerns Telecom Engineering, New Generation Network (NwGN), Network Load Balancing, and Routing Techniques.

Chien Trinh Nguyen is a Professor of Electro-Communication Engineering at the Posts and Telecoms Institute of Technology (PTIT), Hanoi, Vietnam. He received his B.Eng. from the University of Electro-Communications, Odessa, Ukraine, in 1989, and an M.E. and a Ph.D. from the University of Electro-Communications, Tokyo, Japan, in 1999 and 2005, respectively. He joined the Research Institute of Posts and Telecoms (RIPT), Hanoi, Vietnam, in 1990, where he was engaged in the research and development of digital switching systems, packet switched networks, network planning, design, and optimization. Since 2013, he has been the director of the Telecom Department at PTIT. His research interests include new generation network (NwGN) technologies, network traffic analysis and modeling, network resource allocation, network QoS, and network security.