International Journal of Computing and Network Technology
ISSN (2210-1519)
Int. J. Com. Net. Tech. 4, No.3 (Sep-2016)

Improved QoS Routing Model with Local Information of Network State

Tran Minh Anh and Nguyen Chien Trinh

Posts & Telecommunications Institute of Technology, Hanoi, Vietnam

Received: 5 May 2016, Revised: 12 Aug. 2016, Accepted: 22 Aug. 2016, Published: 1 Sep. 2016

Abstract: QoS (Quality of Service) routing algorithm based on localized information has recently been proposed as an alternative approach to the traditional QoS routing algorithms that use global state information. By this method, the localized algorithm, using information collected from source node, helps flow routing better and assures QoS more flexibly for network. This kind of scheme will be able to become a new solution for satisfying the higher and higher demand of telecom services in the near future. In this paper, we introduce a new routing algorithm, an effective localized QoS routing which can be used to assure the quality of network, and show our simulations that perform better results than other routing algorithms.

Keywords: algorithm, information, localized, QoS routing

1. INTRODUCTION

Nowadays, telecom network enlarges very strongly and the assuring of quality for large-scale networks becomes very challenged. Therefore the approach of using local information at source node about congestion probability, network statistics … to build sets of paths available for routing is the way of effective conveying information in network, especially in the very large network. When local information is used, it may improve the overall performance of network and it has been demonstrated that this technique is simpler and better than global QoS routing schemes which update the network state information periodically by a link-state algorithm and maintain it up-to-date all the time. That will lead to a large communication overhead, significantly affects scalability, the inexact of global state and the out-of-date information due to large update intervals.

Localized QoS routing method otherwise tries to avoid these problems by routing based on local information. It means it performs flow routing by using the localized view of the network QoS state. In this approach, each node builds and maintains a predetermined set of candidate paths to each possible destination and routes flows along these paths. This localized method will open a new approach to assure QoS for network in the near future.

In this paper, we will introduce a new routing algorithm (we call *bdr*: bandwidth-delay based routing) which is a localized QoS routing algorithm which uses the two parameters of QoS (bandwidth and delay) as criteria for routing, as well as uses a path index based on the probability of flow transmit success. We compare and realize better performance against the Quality Based Routing algorithm (cbr) proposed in [6], the Proportional Sticky Routing (psr) algorithm proposed in [7, 8] and the traditional global QoS routing algorithm Widest Shortest Path (wsp) when performing them using the same types of topology, traffic patterns and under the same range of traffic loads.

2. RELATED WORKS

There are a lot of researches for QoS routing recently which have been published on many different areas as [1-5] and wherein. The parameters of QoS like: Bandwidth, Delay, Delay Jitter … are now more and more important for telecom applications. After some recent researches, localized QoS routing is a quite new approach in the telecom networks, and the idea of routing based on local information has been used in many dynamic routing schemes. One of these schemes is the scheme of Localized Credit Based Routing algorithm (*cbr*), see [6].

The CBR uses a simple routing procedure to route traffic across the network by using crediting scheme for each candidate path that rewards a path upon flow acceptance and penalizes it upon flow rejection. The larger path credits, the larger chances for selection. The CBR algorithm keeps updating each path's credit upon

E-mail address: anhtm.dng@vnpt.vn, trinhnc@ptit.edu.vn

flow acceptance and rejection and it does not compute a flow proportion. It is also keeps monitoring the flow blocking probabilities for each path and conveys the data to the credit scheme to use it in path selection.

The CBR predetermined a set of candidate paths R between each pair of source and destination where $R = R_{min} \cup R_{alt}$ (R_{min}: a minimum hop set and R_{alt}: an alternative paths set). The CBR selects the largest credit path P.credits in each set, minimum hop (minhop) paths set R_{min} and alternative paths set R_{alt} upon flow arrival. The flow is routed along the minimum hop path that has the largest credit P_{min} which is larger than the alternative path that has the largest credits P_{alt} following this formula:

$$P_{min}.\text{Credits} \geq \Phi \text{ x } P_{alt}.\text{Credits} \text{ , where } \Phi \leq 1 \quad (1)$$

Otherwise, P_{alt} will be chosen. The CBR uses the parameter Φ as a system parameter that controls the usage of alternative paths. The CBR also uses blocking probability in crediting schemes to improve the performance of the algorithm. The path credits are incremented or decremented upon flow acceptance or rejection using statistics of the path blocking probability.

Besides, the CBR uses a MAX_CREDITS system parameter to determine the maximum attainable credits for each path by computing the blocking probability.

$$0 \leq P.\text{Credits} \leq \text{MAX\_CREDITS} \quad (2)$$

The CBR algorithm records rejection and acceptance for each path and uses a moving window for a predetermined period of M connection requests. It uses 1 for flow acceptance and 0 for flow rejection, dividing the number of 0's by M to 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, then 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 [7, 8] 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_{min} and the set of alternative paths P_{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_{eli} 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_{eli}.

A parameter γ_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 γ_r, that path becomes ineligible. When all the paths in the set P_{eli} become ineligible, the cycle will end, and the next cycle begins.

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. However, due to delivering the load proportionally to the paths in the determined set, it will commit a bit high congestion when the load is too fragmented at the destination.

This scheme determines paths for flows eligibly, but it must do a large number of computation for set of paths at first, therefore, sometimes it becomes critical effectively.

3. PROPOSITION AN APPROACH TO QOS ROUTING BASED ON LOCALIZED INFORMATION

A. *Methodology:*

At present, when Internet develops greatly and telecom services unify in using the same network background, these services, especially online or interacting services, demand many metrics of QoS such as Bandwidth, Delay, Delay Jitter, Packet Loss and so on. We realize that the metrics of Bandwidth and Delay which are most often demanded by customers, such as customers of high quality services like: Video on Demand, Tele-Conference … Therefore, other than above methods (they only use bandwidth or shortest path), we propose to choose two main metrics of QoS: Bandwidth and Delay to be the criteria of choosing path for shortening the computation, and we call our scheme as Bandwidth-Delay based Routing or *bdr*.

B. *Describing the algorithm of bdr:*

Like *cbr*, *psr* … our algorithm, *bdr*, also requires every node to maintain a predetermined set of candidate paths R to each possible destination, but unlike *cbr*, *psr* which distinguish between two types of paths *minhop* paths and *minhop+1* paths, *bdr* only builds and keeps track for only

one set of candidate paths R (the result of finding shortest paths).

In our scheme, every path P∈R associated with a variable P.Criteria[1,2]=P.[Bandwidth,Delay] with the index of that path, called Path_idx which is the probability of flow transmission on that path, counted by total flow accepted on that path/total flow transmitted between source and destination. The way to build Path_idx will be discussed later. And, to choose path, the set R (all of candidate paths) will be ranged for Path_idx, called Range R(Path_idx).

Here is the way that *bdr* works: Upon a flow's arrival:

1- Select the path P with max Path_idx (first place in set R after ranging)

2- Check the demand of the flow from SLA (service level agreement) and use it for choosing path, we call that is SLA.[Bandwidth,Delay], or SLA.

3- Compare P.Criteria and SLA (the way to compare will be discussed in part 3.3).

- If P.Criteria ≥ SLA, the P will be chosen.
- If P.Criteria < SLA: select the next P in the array of R (the max Path_idx of the rest of R)
- The loop will be done until finding out the path has maximum of Path_idx and P.Criteria ≥ SLA.
- If no path has quality satisfying SLA, the arriving flow obviously is cancelled.
- Increase Path_idx if route P is accepted; decrease Path_idx if route P is unaccepted accordingly.

Flow chart of all steps:

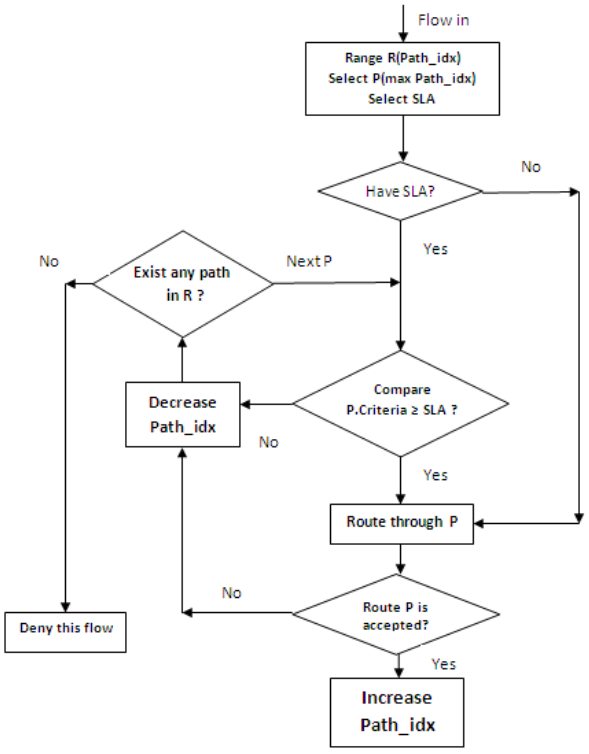

Figure 1. Flow chart of *bdr*

Unlike cbr changes P.Quality as penalty for path, bdr just changes the index Path_idx of this path. The index Path_idx is the probability of success that the path when it is accepted for a flow-in. When the flow is accepted along the selected path, the path index Path_idx is updated with the increasing an amount.

With Path_idx: Path_idx is initially set to one for the first time. When the path is accepted and the value of that path criteria is greater than the value of requested quality (SLA), this indicates that the candidate path has good quality, and Path_idx of that path increases an amount (by the formula below), since the same candidate path may be selected repeatedly if the value of path quality is greater than the requested quality.

Path_idx = (Last Path_idx + 1/∑(times be requested))

(where Path_idx don't exceed 1, if Path_idx exceeds 1, Path_idx gains 1)

When the path is not accepted, another path is chosen, and the value of Path_idx of that path decreases by formula below:

Path_idx = (Last Path_idx – 1/∑(times be requested))

(where Path_idx is not below 0, if Path_idx is below 0, Path_idx gains 0)

When the number of flows requested between a specific pair (source-destination) exceeds a number T, the parameter Path_idx will have the formula:

Path_idx = (Last Path_idx + 1/T)

(Path_idx don't exceed 1, if Path_idx exceeds 1, Path_idx gains 1) and

Path_idx = (Last Path_idx – 1/T)

(Path_idx is not below 0, if Path_idx is below 0, Path_idx gains 0)

where T: a parameter is set first, depends on the capacity of network. Last Path_idx: the value of Path_idx before next flow comes to source node.

After that, when the following flow comes, the algorithm will use this Path_idx to be criteria to choose path, and next loop re-begins. Therefore, after one flow processed, Path_idx changes accordingly to the success/fail of that path, and probability of that path changed correspondently. It affects to the probability of being chosen for the next.

C. 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 relevant and popular way which is used will be discussed below:

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: P.Bandwidth≥F.Bandwidth and P.Delay≤F.Delay)

where Bandwidth: the minimum residual bandwidth of any link on path and Delay: the sum of all delay of propagation from all link on this path).

The procedure is as follows:

```
PROCEDURE Compare(p,f)
If p.Bandwidth ≥f.Bandwidth
    If p.Delay <=f.Delay
        p is chosen
    Else
        p is discarded
Else
    p is discarded
END PROCEDURE
```

Initialize
Building R
Set Path_idx=1,∀ P∈R
Set T=1000
bdr
1. Range Path_idx for flow-in
2. Set P.success = false;
3. Set SLA for flow-in
4. P=first{P.Path_idx: P∈R}
5. Do while !(P.success or R(end))
6. if(P.Criteria≥SLA)
7. Route flow along path P
8. if P is accepted
9. {Compute P.Path_idx (increase Path_idx(<=1))
11. P.success=true}

12. elseif {
13. P= next{P.Path_idx: P∈R}
14. Compute P.Path_idx (decrease Path_idx(≥0))
15. endif}
16. Else
17. P= next{P.Path_idx: P∈R }
18. Compute P.Path_idx (decrease Path_idx(≥0))
19. Endif
20. Loop
21. END.

Figure 2. The *bdr* pseudo code

There are also other ways to compare as in [11] and therein. For example:

- Estimating and using probability of some metrics.
- Using the mixed metric.
- 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, in spite of some ways effective to use in routing with some criteria.

D. The pseudo code of bdr:

4. PERFORMANCE EVALUATION

In this section, we realize the performance of the bdr scheme and compare it with the cbr and psr schemes. Besides, we also compare with the global QoS routing scheme widest shortest path (wsp) which searches for a feasible path with minimum hop count. All the

experiments will be set in the same condition. Next, we analyze the results of our simulation model and performance metrics of the three schemes.

A. Simulation Model

We use simulator based on OMNeT++ [12], 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 as the simulation in [6-8], described as follow: links are all bidirectional with the same capacity C = 20Mbps in each direction and the same value of delay D = 20ms, flows arrive to each source node according to a Poisson process with rate λ and destination nodes are selected randomly (each node is capable of being source and/or destination), flow duration is exponentially distributed with mean $1/\mu$, flow bandwidths are uniformly distributed within [0.5-4MBytes], and the required value set for Delay of each flow is randomly distributed between 20ms and 250ms.

As analyzed in [9-10], 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=18, L=60, h=2,36, $1/\mu$= 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 λ according to experiments of from low loads to high loads. The simulated network is as follows:

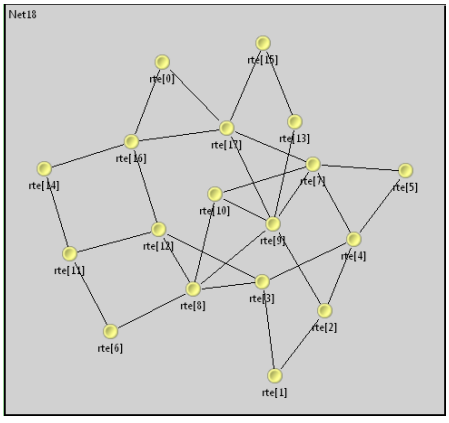

Figure 3. The *bdr* pseudo code

To compare with other schemes, we choose flow blocking and bandwidth blocking probabilities as criteria as well as the simulation in [6-10]. The blocking probabilities are calculated based on the most recent 10,000 flows. The time of simulation is set about of 20 minutes, equivalent of more than 2.5 million of flows emitted. Then, the standard overall flow blocking probability is defined as:

$$\text{Flow Blocking Probability} = |B|/|T| \quad (3)$$

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

$$\text{BBP} = \sum(\text{bandwidth of } |B|)/\sum(\text{bandwidth of } |T|) \quad (4)$$

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

B. Simulation Results

With the results of flow blocking probability, we collect information from the simulation and compare with the ones of other schemes, as shown in Fig 4 to Fig 5.

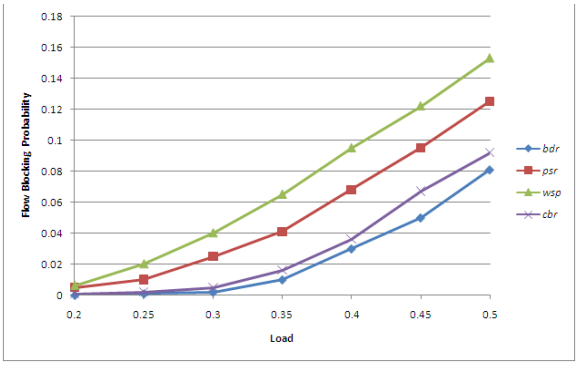

Figure 4. Flow Blocking Probability

Fig. 4 and Fig. 5 show the performance of *bdr* against *cbr*, *psr* and *wsp* in terms of flow and bandwidth blocking probabilities under load ρ varies from 0.2 to 0.5. From these values, we 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.

When ρ is high (more than 0.3), some differences reveal. Many flows drop or/and fail to get destination node, then flow blocking probability grows rapidly, involving bandwidth blocking probability is high at the same time as view in Fig. 4 and Fig. 5.

The reason of this: In the case of the *bdr* scheme, paths are selected based on probability of predetermined paths, then the index of that path increases assures that this path is good and may support relatively the next flow. It means that it has available bandwidth in the links on the path selected. Otherwise than *cbr*, *bdr* uses Delay as a criterion to compare, so all flows which are chosen, almost satisfy the demand of Delay at destination, so it helps diminishing flow blocking probability of that path,

and diminishing the value of Bandwidth Blocking Probability of those flows as well.

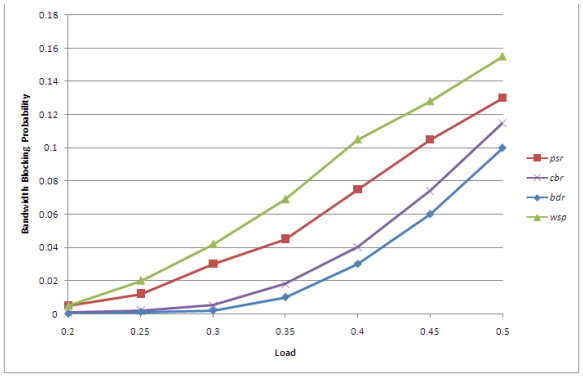

Figure 5. Bandwidth Blocking probability

Moreover, the setting index for path chosen 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* and *wsp* as well.

Next experiments, we collect information about metrics of Average End-to-End Delay in the different load (with load ρ=0.2 and 0.5). The required value set for Delay of each packet is randomly distributed between 20ms and 250ms. We calculate the results of our case *bdr* with the case of *wsp* (using Dijkstra algorithm [19] with weighted links to route). And the results are shown in Fig. 6.

From the Fig. 6, we can see that when the number of flows increases, the Average End-to-End Delay will keep a stable value as shown and the *bdr* expresses the more efficiently. The margin between two cases is slightly low at ρ=0.2, but when the load ρ = 0.5, it becomes higher as shown below.

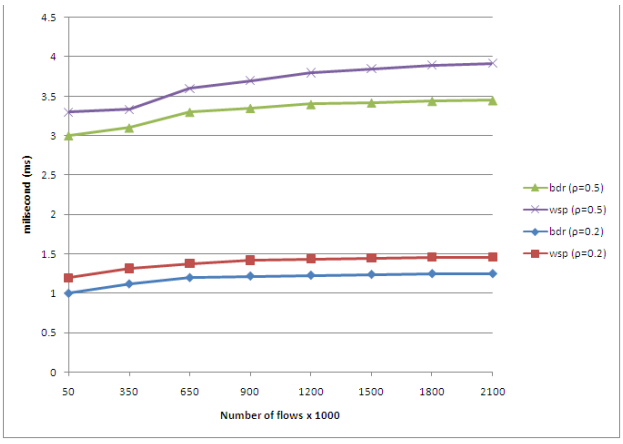

Figure 6. Average End-to-End Delay of flows when ρ=0.2 and 0.5

It means that with high load, the congestion happens more frequently, so, this value is higher. In our case, the flows change path more frequently based on the index Path_idx. When congestion happens, the index of the regular path diminishes, then, our case changes path. Therefore, the average End-to-End Delay is better than case of *wsp* as well.

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

C. *Complexity and overhead*

The case of *wsp* uses the algorithm Dijkstra, like almost global QoS routing algorithms, takes at least O(NlogN+E) time, where N is the size of the network measured in the number of nodes, and E is the number of links (edges).

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 that is |R|.

In the *cbr* and *psr* algorithm, the path selection is an invocation of a weighted-round-robin like path selector (wrrps), whose worst case time complexity is O(|R|) as [6], similarly *bdr* 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).

Therefore, with communication overhead, *bdr* 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 localized of our case 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 model to choose path using information collected locally at source node. We also propose the flow chart and pseudo code of the algorithm which uses the two QoS metrics (bandwidth and delay) as criteria to route flows through network. Many experiments have been done to compare between its performance against the cbr, psr algorithms, and also against the wsp algorithm; and have showed a comparable performance with better blocking probabilities, better time complexity and lower communication overhead.

As part of future work, we will survey this QoS routing algorithm which uses more QoS parameters to compare, loss packet or delay jitter and so on. It will of course make the routing more flexible in spite of making a lot of computation and complexity.

And finally, we research the way of building set of candidate paths for this algorithm. In spite of wasting time of overhead, the more effective set of paths will make the algorithm operate more exactly and more reliably.

REFERENCES

[1] C. Pornavalai, G. Chakraborty, N. Shiratori, QoS based routing algorithm in integrated services packet networks, in: Proceedings of the IEEE ICNP, 1997.

[2] S. Chen and K. Nahrstedt, "An Overview of Quality-of-Service Routing for the Next Generation High-Speed Networks: Problems and Solutions," IEEE Network, special issue on transmission and distribution of digital video, vol. 12, pp. 64-79, 1998.

[3] R. Guerin and A. Orda, QoS based Routing in Networks with Inaccurate Information: Theory and Algorithms, IEEE Transaction on Networking, 7(1999), pp.605-614.

[4] Z. Whang and J. Crowcroft, Quality-of-Service routing for supporting multimedia applications, IEEE J. Select. Areas Commun, 14 (1996), pp. 1228-1234.

[5] Q. Ma and P. Steenkiste, Quality-of-Service routing for traffic with performance guarantees, 5th Int. IFIP Workshop on QoS, IEEE, New York, NY, 1997, pp. 115-126.

[6] S. Alabbad, M. E. Woodward. "Localized Credit Based Routing: Performance Evaluation Using Simulation", Proc. of IEEE 39th Annual Simulation Symposium, Huntsville, Al. USA April 2-6 2006.

[7] S. Nelakuditi, Z. L. Zhang, R. Tsang and D. Du, On selection of candidate paths for proportional routing, Computer networks, 44 (2004),pp. 79-102.

[8] S. Nelakuditi, Z. L. Zhang, R. Tsang and D. Du, Adaptive Proportional Routing: a Localized QoS Routing Approach, IEEE/ACM Transactions on Networking 10 (2002), pp. 790-804.

[9] A. Shaikh, J. Rexford, K. Shin, "Load-Sensitive Routing of Long-Lived IP Flows", ACM SIGCOMM 1999.

[10] A.Shaikh, J.Rexford, K.G.Shin. Efficient Precomputation of Quality-of-Service Routes. Proc. IEEE NOSSDAV 98, July, 1998.

[11] Wei Sun, QoS/Policy/Constraint Based Routing, The Ohio State University, 1999.

[12] A. Varga, The OMNeT++ Discrete Event Simulation System, the European Simulation Multiconference, Prague, Czech Republic, 2001

Tran Minh Anh received his B.S. degree in Electronics and Telecommunication Engineering from Danang University of Technology in 1995. In 2001, he received his M.S degree in Telecommunication Engineering from Hanoi University of Science and Technology. His research interests include New Generation Networks (NwGN) Trends, Telecom Engineering, Network Load Balancing, Routing Technique.

Nguyen Chien Trinh is a Professor of Electro-Communications Engineering at the Posts and Telecommunications Institute of Technology (PTIT), Hanoi, Vietnam. He received his B.E. degree from the University of Electro-Communications, Odessa, Ukraine in 1989, received his M.E and Ph.D. degrees in University of Electro-Communications, Tokyo, Japan, in 1999 and 2005, respectively. His research interests include new generation networks (NwGN) technologies, network traffic analysis and modeling, network resource allocation, network QoS, network balancing and network security.