Influence of Link State Updating on the Performance and Cost of QoS Routing in an Intranet

Zhansong Ma, Peng Zhang, Raimo Kantola

Networking Laboratory of Helsinki University of Technology
Otakaari 5A, Espoo, FIN 02015, Finland
Email: {zhansong, pgzhang, kantola}@tct.hut.fi

***Abstract*-QoS routing has been regarded as an essential enhancing mechanism for providing differentiated QoS in the IP networks. QoS routing needs link state updating algorithms for acquiring accurate link state information (e.g., available link bandwidth) to compute routes. However, an unsuitable link state updating also brings excess computation and communication cost so that seriously degrades the performance of QoS routing. In this paper, we present four link state update algorithms for QoS routing in an Intranet, i.e., period based, threshold based, equal class based, and unequal class based updating. By doing extensive simulations, we investigate and compare their impact on the performance and cost of on-demand QoS routing in an intranet. Our simulation results prove that QoS routing can achieve both high performance and low cost under the careful selection of link state updating algorithms and their parameters.**

I. INTRODUCTION

With the success of the Internet in recent years, today's Internet armed with best-effort routing is being expected to support various services, not only the traditional services (e.g., email, FTP) but also the upcoming high speed and real-time services (e.g., audio/video real-time transmission, virtual private network). The latter services represent much different traffic characteristics from the former services in terms of bit-rate and burst, and they require fixed assurance of Quality of Service in the duration of transmission. However, the current Internet does not support explicit guarantees for these services. As a result, a need for a high performance network emerges.

So far, great efforts have been put forward to provide guarantees for specific services or customers, e.g., Integrated Services with the Resource Reservation Protocol and the Differentiated Services Architecture. MultiProtocol Label Switching is a forwarding scheme, which can be used together with DiffServ to provide better QoS.

QoS routing computes paths that are subject to QoS requirements, and at the same time aims at achieving high efficiency in network resource utilization. It has been recognized as an important part in the evolution of QoS-based service offerings in the Internet and more and more attention has been attracted to the field of QoS routing. Some research results [1][2][3] have pointed out its potential benefits:

- enabling creation of virtual circuit-like services over IP networks;
- improving user satisfaction by increasing chances of finding a path that meets the QoS requirements;
- improving network utilization by finding alternate paths around congestion spots.

However, those benefits do not come for free, a variety of extra costs comes along: the cost of deploying QoS routing protocol, which may include software cost, operation cost and maintenance cost, the cost of incurring potentially higher communication, processing and storage overheads. As a result, one major concern facing the deployment of QoS routing is its feasibility that argues its benefits and its drawbacks of increasing the overall cost. Among those added costs, the processing cost of link state updates is regarded as a major cost contributor in QoS routing [4][5] and it also has significant influence on the performance of QoS routing. Different link state updating (LSU) algorithms have different performance and cost characteristics. Understanding their specific behavior in different network conditions is helpful when considering the implementation of QoS routing.

In this paper, we investigate how different LSU algorithms impact the performance and cost of on-demand QoS routing in an intranet by using a QoS Routing Simulator (QRS) [6]. The rest of this paper is organized as follows. In section II, we first discuss the relationship between link state updating and the performance and cost of QoS routing, then present four LSU algorithms implemented in QRS. In section III, we present our simulation environment and simulation results. Section IV is our conclusions and suggestions for future study.

II. LINK STATE UPDATE ALGORITHMS

One basic requirement of QoS routing is tracking the network state information so that the state information is available to the path computation. The standard link state protocols and distance vector protocols update state information periodically. The main disadvantage of this approach is that it can not ensure timely propagation of significant changes, and therefore can not ensure providing accurate information for path computation. Ideally, nodes should be able to catch the instant view of the network, which theoretically could be ensured by instantly updating the state information. But if the state information changes very quickly from time to time, updating state information for each change will cause a great burden for the network links and routers−consuming much network bandwidth and routers' CPU cycles. One way to solve this problem is to set a threshold to distinguish significant changes from minor changes. And the state information updating is triggered when a significant change occurs. Alternately, network

resources can be partitioned into ranges or classes, the state information updating is triggered for each class boundary crossing. Such methods provide some control on the tradeoff between information accuracy and volume of updates. However, periods of rapid traffic fluctuations may trigger frequent updating and, as a result, cause transient control overloads. To alleviate this problem, a hold-timer can be invited to complement the threshold and the class based triggering methods to enforce a minimum spacing between consecutive updates. In this paper, we study the following four LSU algorithms: period based (PB), threshold based (TB), equal class based (ECB), and unequal class based (UCB) updating.

The basic idea of the PB algorithm is to update the whole topology periodically. A constant timeout is set to determine when the network states should be updated. This algorithm provides direct control over the communication overhead, but does not ensure timely propagation of significant changes especially when the timeout is assigned a big value.

The idea of TB, ECB and UCB algorithms is that the scope of a node's update extends to all its incident links, that available bandwidth values for all the interfaces of the node are advertised even when the update is triggered by just one link. It is also in compliance with the behavior of routing protocols such as OSPF [7] that only generate LSUs on all the links attached to a node. In addition, TB, ECB and UCB attempt to trigger an update only when the current available bandwidth of a link differs significantly from the previously advertised value.

- TB: In this algorithm, a constant threshold value (*th*) is set. For an interface *i* of a node, let bw_i^o be the last advertised value of available bandwidth, and bw_i^c is the current value, an update is triggered when $(|bw_i^o - bw_i^c| / bw_i^o) > th$ for $bw_i^o > 0$. For $bw_i^o = 0$, an update is always triggered. This algorithm tends to provide more detailed information when operating in the low available bandwidth range and becomes progressively less accurate for larger value of available bandwidth.
- ECB: In this algorithm, we set a constant *B* that is used to partition the available bandwidth operating region of a link into multiple equal size classes*: (0,B),(B,2B), (2B,3B),…,etc*. An update is triggered when the available bandwidth on an interface changes so that it belongs to a class that is different from the one to which it belonged at the time of the previous update. It has the same degree of accuracy for all ranges of available bandwidth.
- UCB: In this algorithm, we set two constants *B* and *f (f>1)* that are used to define unequal size classes: *(0,B), (B,(f+1)B), ((f+1)B, $(f^2+f+1)B$), ($(f^2+f+1)B$, $(f^3+f^2+f+1)B$),…*, etc. Unlike the equal class based algorithm, the class sizes grow geometrically by the factor of *f*. An update is triggered as before, i.e., when a class boundary is crossed. This policy has fewer and larger classes in the high available bandwidth operating region and more and smaller classes when available bandwidth is low. Consequently, it tends to provide a more detailed and accurate state description for the low bandwidth region.

III. SIMULATION STUDY

A. QoS Routing Simulator

We had developed a discrete-event QoS routing simulator (QRS) from the Maryland Routing Simulator (MaRS) [8] by adding and modifying some components for handling QoS routing. Particularly, we implemented a QoS routing protocol called QOSPF which is extended from the SPF component in MaRS and installed four LSU algorithms described in section II. We created a Realtime Traffic Source/Sink component that is used to generate real time traffic. QRS can be used to investigate issues of QoS routing in the network at intra domain level.

B. Metrics

Performance: We use the average network throughput achieved by real-time traffic with bandwidth requirements to represent the network performance. To get the average network throughput, we log the number of received packets in real-time traffic sinks during the simulation, then calculate the average throughput: $\Sigma_i(N_i*L_i)/t$, where *N* is the number of packets which are received by real-time traffic sinks, *L* is the size of the packet, *t* is the simulation time and *i* represents the type of the packet.

Cost: We use total processing time consumed by QOSPFs during the simulation time to represent the cost of QoS routing. To get the cost of QoS routing, we log the time consumed by QOSPF in every node during the simulation time, and then simply calculate the sum.

The processing time of each action of QOSPF in a node is set as shown in Table 1.

Table 1 Cost of each QOSPF action

No	Cost(us)	Action Description
1	1500	Find a next hop that can accept the required bandwidth
2	100	Check a message from RSVP and decide next step
3	1500	Compute the QoS path
4	500	Update the local topology database
5	200	Broadcast the link state information
6	100	Broadcast a message packet
7	1000	Compute normal routing table for best effort traffic

The time consumed by action 1, 3 and 7 specified in Table 1 should not be constants. In reality they are functions of the network size. For simplicity in our work, we define them to be constants and thus limit the amount of programming work and simplify the observing of the impact of LSU on QoS routing cost.

C. Simulation Environment

We aim to evaluate the performance and cost of QoS routing under various network environments. Particularly, we focus on a certain period of time of operational network, in which a number of requests are handled. By recording the average network throughput and the consumed time, we can observe how the performance and cost changes with different LSU algorithms with different parameter values.

We use matrix topologies (Fig. 1) and ISP topology (Fig. 2) to simulate QoS routing. In all topologies, all link propagation delays are 1 millisecond, all links are symmetric

and have the same bandwidth of 6Mb/s, and all nodes have adequate buffer space for buffering packets awaiting processing and forwarding. Besides, none of the links fail during simulation.

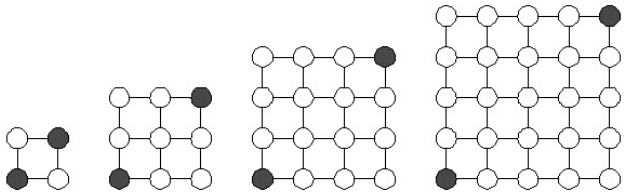

Fig. 1 Matrix topologies

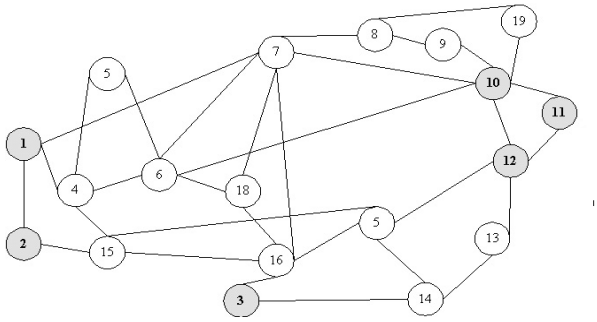

Fig. 2 ISP topology

We use different algorithms as described in section II and combinations of the LSU algorithms with a hold-timers. The routing algorithm used in all simulations is the lowest cost algorithm [6], the link cost is calculated by a hop-normalized delay cost function [8].

Real-time traffic (RT) is modeled in terms of requests for setting up connections with specific bandwidth requirements. A request is characterized by its source, destination, requested bandwidth, active period (ON), inactive period (OFF), etc. [9].

We use simple traffic (ST), FTP and Telnet to model traffic without resource reservation requirements as the background traffic as opposed to RT. Background traffic is installed to be able to fill all incident links of the concerned nodes when there is no RT.

All simulations run 100 seconds.

D. Simulations in Matrix networks

For each matrix topology in Fig. 1, we install 27 RT pairs between the diagonal nodes (black nodes). All pairs have the same source and destination. Each flow rate is set to 0.5 Mb/s. The ONs and OFFs of flows are randomly set to a value from 0.1s to 0.3s. Then the average workload of the RT pairs in the network is about 7Mb/s.

We show the simulation results in Fig. 3 and Fig. 4. The results show that no matter which kind of LSU algorithm is used, the cost of QoS routing increases rapidly along with the increase of network size. The result also show that the way of different LSU algorithms effecting on performance and cost differ with network size. For example, from Fig. 3 and Fig. 4, we can see that when network size is 2*2 and 3*3, PB (100ms) involved simulation provides the best result with least cost and best performance comparing with other three algorithms. However when network size reaches 4*4, 5*5, neither best performance nor least cost is produced by PB, in particular, it is the most expensive one under size 5*5.

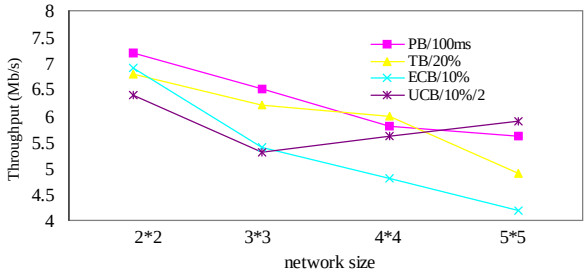

Fig. 3 Performance in matrix topologies under different LSU algorithms

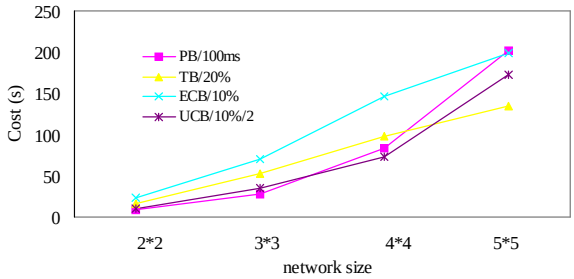

Fig. 4 Cost in matrix topologies under different LSU algorithms

Another phenomenon of interest to us is that even though the cost increases with network size growing in all simulations, the degree of network size impact is significantly different with different algorithms. Fig. 4 shows TB/20% involved simulation presents the modest curve , which means that comparing with other three algorithms, TB/20% is least sensitive to network size under the simulation environment we set.

Moreover, the results show that the performance generally has a modestly decreasing trend with network size increasing. A special case is from UCB involved simulation from which the best performance result is under network size 2*2, then 5*5, 4*4, the worst one is under the size 3*3. We repeat the simulation several times with UCB algorithm, but the results we get are always different. At this stage, we regard that this presents the complexity of UCB which is operated by two variables B and f instead of only one as in other algorithms. Further study is needed in order to get more knowledge of the characteristics of this algorithm.

E. Simulations in ISP network

In the ISP network as shown in Fig. 2, we configure: 7 RT pairs from node 1 to each of the nodes: 10, 11, 12; 7 RT pairs from node 2 to each of the nodes: 10, 11, 12; 7 RT pairs from node 3 to each of the nodes: 10, 11, 12. Totally there are 63 RT pairs. From the ISP network, we can see that the minimal cut (7-8, 7-10, 6-10, 5-12 and 13-12) has five links with the total capacity of 30Mb/s(5×6Mb/s). Obviously, the total network throughput achieved by RT should be at most 30Mb/s.

All RT pairs' ONs and OFFs are set randomly from 1s to 3s and 0.1s to 0.3s respectively. If a request of connection setup fails, it will re-request after 100ms. We construct two different traffic models. One is called uniform traffic (UT) model in which all RT pairs have the same flow rate, the other one is called non-uniform traffic (NT) model in which not all RT pairs have the same flow rate. For the UT, the workload of every real-time traffic flow is set to be 0.5Mbps. The average total workload is about 27Mb/s. For the NT, the workload of each real-time traffic flow is distributed randomly from 0.1Mbps to 3Mpbs. The average total workload is about 28Mb/s.

We repeat the simulation with different LSU algorithms and sort the results into a chart.

1. Simulation results under different PB values

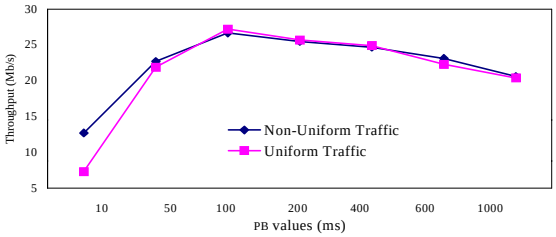

Fig. 5 Performance under NT/UT with different PB values

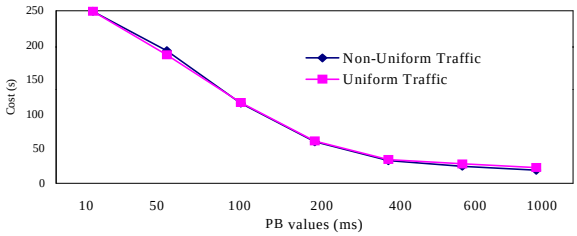

Fig. 6 Cost under NT/UT with different PB values

The results recorded in Fig. 5 and Fig. 6 show that small PB values (e.g.,10ms and 50ms) lead to bad performance and high cost. This is because the small PB values lead to high frequency of LSU which directly leads to a high cost. On the other hand due to the propagation delay of broadcasting LSU information, the high frequency of LSUs causes the state information to become unstable for path computation, which can be the reason of bad performance.

However, referring to Fig. 5, with a suitable PB value (100ms), the performance is almost perfect and achieves the average rate of RTs. Furthermore, the QoS routing cost drops significantly with the increased PB values (e.g., from 100ms to 400ms) while the performance drops smoothly. Further increase of the PB value (e.g., from 400ms to 1000ms) causes both cost and performance to drop smoothly.

2. Simulation results under different TB values

Fig. 7 and Fig. 8 show that with the TB algorithm, different traffic models result in different performance and cost. The cost in UT is higher than in NT, but there is no exact pattern to compare their performance differences.

Under both UT and NT models, the cost drops sharply with the increase of the TB value, but the best performance on the other hand is not associated with the highest cost, it happens when 20% change is considered significant in the available bandwidth (20% TB).

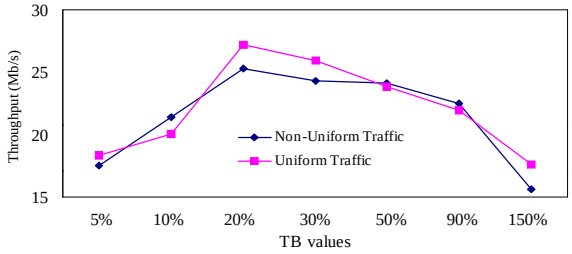

Fig. 7 Performance under NT/UT with different TB values

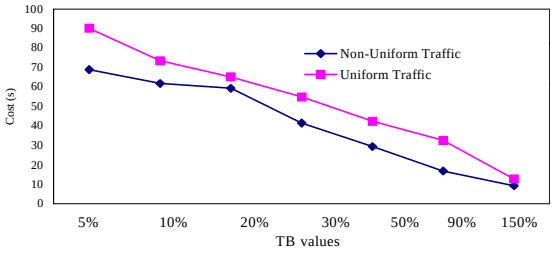

Fig. 8 Cost under NT/UT with different TB values

We also did the simulations with different ECB values, i.e., 5%, 10%, 20%, 25% 50% 100% 200%. The results are similar to the above results shown in Fig. 7 and Fig. 8. But the best performance is only 23Mb/s, less than 27Mb/s- the performance by using TB with the value of 20%.

3. Simulation results under different 20% TB hold-timers

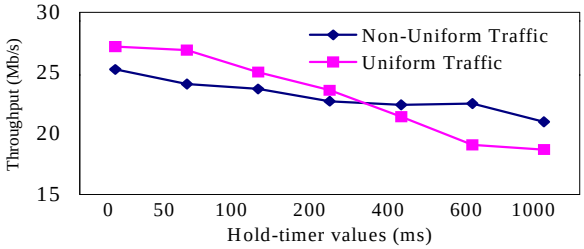

Fig. 9 Performance under NT/UT with different 20% TB hold-timers

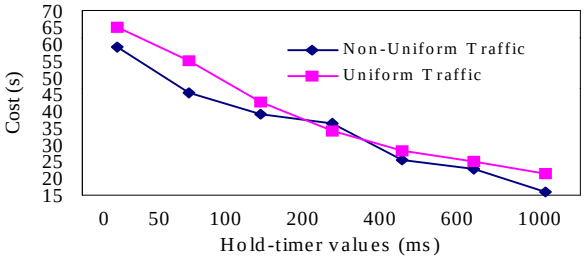

Fig. 10 Cost under NT/UT with different 20%TB hold-timers

Fig. 9 and Fig. 10 imply that with 20%TB algorithm, the hold timer value has great influence on cost, but minor impact on performance. With the increase of hold-timer value, the cost drops sharply under both NT and UT models. This result reflects the role of hold-timer in the cost of QoS routing.

4. Simulation results under different UCB values

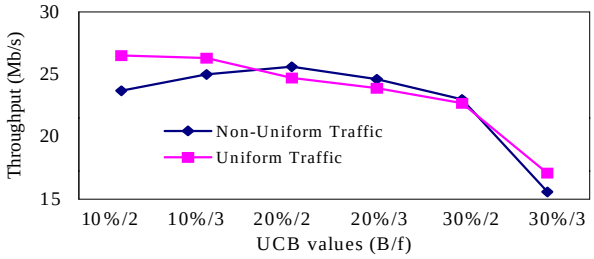

Fig. 11 Performance under NT /UT with Different UCB Values

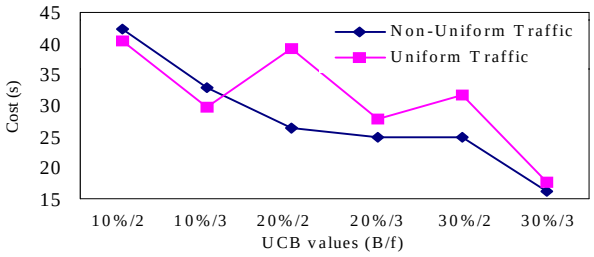

Fig. 12 Cost under NT /UT with Different UCB Values

Fig. 11 and Fig. 12 show that with the UCB algorithm, changing the values of B and f, both the performance and cost show irregular changes. We regard that this shows again the complexity of the UCB algorithm with two controlling parameters.

By further comparing the UCB with the TB, we observe a similar performance result. (For example the performance with TB 20% in Fig. 7 and UCB 10%/3 in Fig. 11), but the cost produced by UCB is much lower than the cost produced by TB (Fig. 8 and Fig. 12). This implies that UCB with two controlling variables is more flexible than TB with only one adjustable factor. We deduce that with the optimum B and f values, UCB will possibly be able to achieve the best result in performance and cost combination comparing with other algorithms. But more study is needed before we could make a more convincing conclusion.

From the above four simulation results, we find the best performance of QoS routing achieved with each LSU algorithm is above 25Mb/s. Since the capacity of each link is 6Mb/s, so this result means that at least 5 paths are used to transport real-time traffic simultaneously during the simulations. By analyzing the ISP network, we are convinced that this result can not be achieved by a best effort routing scheme.

IV. CONCLUSIONS

In this paper, we presented a preliminary simulation-based study of the influence of link state updating on the performance and cost of QoS routing. As results, we found:

- The impact of network size on the performance and cost vary with different LSU algorithms;
- In PB, the cost is reduced with the increase of the PB value. Both very small and very big PB values can result in bad performance;
- In TB, the cost is reduced with the increase of the threshold value; both small and big threshold values can result in bad performance;
- In UCB, the performance and cost are affected by two variables, by setting suitable values, this algorithm can produce a good combination of performance and cost;
- QoS routing cost can be reduced by using the hold-timer.

These results could be useful when considering the implementation of QoS routing in the Internet in which different network domains may use different LSU algorithms according to the specific characteristics of traffic travelling on it.

For further study, more simulations with different network topologies and traffic models that are closer to the real situation are needed. In addition, new LSU algorithms are encouraged.

REFERENCES

[1] Crawley, E. et al., "A Framework for QoS-based Routing in the Internet", RFC 2386, Aug. 1998.

[2] Wang, Z. & Crowcroft, J., "Quality of service routing for supporting multimedia applications", IEEE Select. Area Communication, Vol.14, no.7, Sep. 1997.

[3] Lee, W. C. & Hluchyj, M. & Humblet, P., "Routing subject to quality of service constraints in integrated communication networks", IEEE Networks, Jul./Aug. 1995.

[4] Apostolopoulos, G. & Guerin, R. & Kamat, S., "Implementation and Performance Measurements of QoS Routing Extensions to OSPF", IEEE , Volume: 2 , 1999 Page(s): 680 -688 vol.2.

[5] Apostolopoulos, G. et al., "Intra-Domain QoS Routing in IP Networks: A Feasibility and Cost/Benefit Analysis", IEEE Network Sep./Oct. 1999.

[6] Zhang, P. & Kantola, R & Ma, Z., "Design and Implementation of QRS (QoS Routing Simulator)", SPECTS'2000, Feb. 2000.
URL: http://www.tct.hut.fi/~pgzhang/papers.html.

[7] Moy, J., "OSPF version 2", RFC 2178, Apr. 1998.

[8] Alaettinoglu, C. et al.,"MaRS (Maryland Routing Simulator)-Version 1.0. User's Manual. URL: http://www.isi.edu/~cengiz/publications/.

[9] Zhang, P .& Ma, Z. & Kantola, R., "QoS Routing Simulator: User's Manual (Version 1.1)", Feb. 2000. URL: http://www.tct.hut.fi/~pgzhang/papers.html.