Influence of Link State Update Algorithms on the Cost of QoS Path Computations

Gonzalo Camarillo
Advanced Signalling Research Laboratory
Ericsson, FIN-02420 Jorvas, Finland
Gonzalo.Camarillo@ericsson.com

Abstract

Routers performing QoS routing need to know the state of the links in the network in order to find appropriate paths. The mechanism used to trigger link state updates influences the routing performance a great deal.

Traditionally periodical link updated triggers have been employed. Using alternative link update triggers, such as threshold based or class based, reduces the processing in the routers and increases the performance of the system. This performance can be measured by the throughput of the sources transmitting real time traffic.

1 Introduction

The Integrated services architecture [1] proposes RSVP [2] as the resource reservation protocol. RSVP reserves resources along a path taking into consideration QoS parameters. Before resource reservation can be performed it is necessary to find a path that fulfils the QoS requirements of the end systems willing to communicate. In order to find paths with QoS constraints it is necessary a routing protocol that contains QoS metrics. QOSPF [3] adds some extensions to OSPF [4] to provide QoS path discovery.

QOSPF, as OSPF, is a link state routing protocol. In link state routing protocols every node has a global picture of the network and calculates paths based on it. The state of the links of the network has to be known in order to find an appropriate path. The requirements imposed on the path might be bandwidth, delay or any QoS parameter supported by the routing protocol.

The network conditions are changing continuously and so the state of the links does. The information that routers have must be as accurate as possible in order to perform correct path calculations. There are various algorithms that can be used for this purpose. Periodical updates, threshold based updates and class based updates are studied in this paper.

On-demand path calculation is another option that might be considered, but it does not suit networks with high QoS request rate. The processing overhead is too large to be implemented successfully in networks with a high level of QoS traffic.

The rest of this paper is organized as follows. In section 2, different update algorithms are presented. Section 3 describes the simulation environment and the metrics chosen to measure the results. Section 4 outlines the results obtained and their consequences. Finally section 5 concludes the paper.

2 Link state update algorithms

Alternatives to on-demand path calculations are based on path pre-computation [5]. Paths to destinations are calculates independently on the requests for connections. Thus, when a request arrives, it is not necessary to calculate at that point of time a feasible path. The path has been computed before. Different algorithms [6] propose different triggers for the link state updates.

The most simple trigger algorithm consists of employing periodical updates. A timer is defined and every time it expires an update is triggered.

When threshold based update policy is used a threshold is defined. This threshold represents a percentage of bandwidth. If the bandwidth variation in a link since last update goes beyond this percentage a link update is performed.

There are two kinds of class based updates: equal class based and exponential class based. In both the bandwidth of the link is divided into different intervals called classes. If the current available bandwidth changes the class it belongs to an update is triggered. The size of all the intervals is the same in equal class based: (0,B), (B,2B)… In exponential class based the size of the classes grows geometrically by the factor f: (0,B), (B, (f+1)B), ((f+1)B, (f^2+f+1)B)…

3 Simulation Environment and metrics chosen

A simulator has been developed at HUT (Helsinki University of Technology) [7]. QRS [8] (QoS based Routing Simulator) is based on MaRS (Maryland Routing Simulator) [9]. [10] shows how to get and install QRS.

QRS implements the four algorithms described in the previous section: periodical, threshold based, equal class based and exponential class based. The simulations described in section 4 have been carried out using QRS over different topologies: matrixes of different sizes and trees of different sizes. The following figures show the four different topologies employed.

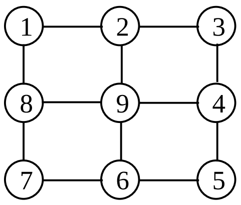

Figure 1 : Matrix with 9 nodes

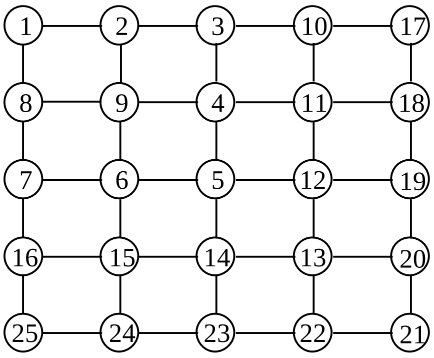

Figure 2 : Matrix with 25 nodes

In the matrix with nine nodes the traffic is generated in node one and terminated in node 5. In the matrix with 25 nodes the traffic flow goes from node 1 to node 21. The configurations files used with QRS can be found at [11] and [12] respectively.

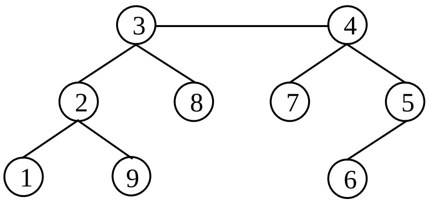

Figure 3 : Tree with 9 nodes

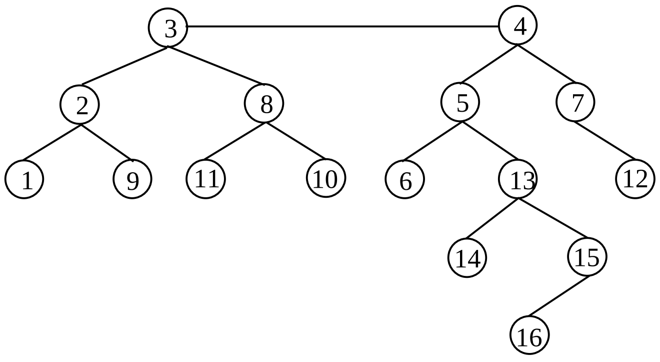

Figure 4 : Tree with 16 nodes

In the tree with 9 nodes there is traffic between nodes 1 and 6. In the larger tree there is traffic between nodes 1 and 16. The configurations files used with QRS can be found at [13] and [14] respectively.

In order to study the performance/cost of the different algorithms throughput and time consumed by QOSPF have been chosen as metrics for these simulations. The throughput is measured counting the number of packets received by the destination node. The cost is the sum of the different times employed by OSPF in each node. Next section shows the result of the simulations.

4 Simulation results

In order to compare the performance/cost of the different algorithms 12 real-time sources has been installed in the network. They generate traffic with different characteristics. The traffic profile is defined in the configuration files pointed in the previous section. A source of best effort traffic loads the network with background traffic.

The parameters used for the algorithms are contained also in the configuration files. Periodical updates were performed every 10 ms. In the threshold based update algorithm 0.1% was used as a threshold. For the class based algorithms 21 classes of size 0.01 were used. The factor ‘f’ of the exponential class based algorithm was set to 2. All the simulations were running for 20 seconds. The following tables show the results of this simulation in different topologies.

Table 1 : Matrix with 9 nodes [15-18], [19-22]

Algorithm	**Throughput Real-time/best effort**	**Cost (µs)**
Periodical	190 / 3002 packets	11415000
Threshold based	1973 / 1424 packets	7677000
Equal class based	2715 / 1702 packets	9859000
Exponential class based	1896 / 1981 packets	7342500

Table 2 : Matrix with 25 nodes [23-26], [27-30]

Algorithm	**Throughput**	**Cost (µs)**
Periodical	165 / 670 packets	31381500
Threshold based	995 / 2269 packets	36024000
Equal class based	712 / 2237 packets	36093000
Exponential class based	1999 / 1633 packets	36073500

Table 3 : Tree with 9 nodes [31-34], [35-38]

Algorithm	**Throughput**	**Cost (µs)**
Periodical	2015 / 598 packets	10149000
Threshold based	1968 / 715 packets	5236500
Equal class based	1954 / 726 packets	6219000
Exponential class based	1995 / 689 packets	4986000

Table 4 : Tree with 16 nodes [39-42], [43-46]

Algorithm	**Throughput**	**Cost (µs)**
Periodical	1525 / 470 packets	16822500
Threshold based	1658 / 945 packets	14284500
Equal class based	1727 / 874 packets	14388700
Exponential class based	1754 / 850 packets	13855500

In the title of every table two references are given. They contain the output of the simulation. The fist reference contains the configuration file with the values of all the parameters. The second reference contains data about the throughput of the system in different moments of the simulation.

The following graphics show the same data as the tables. With this graphical representation it is easier to compare the algorithms.

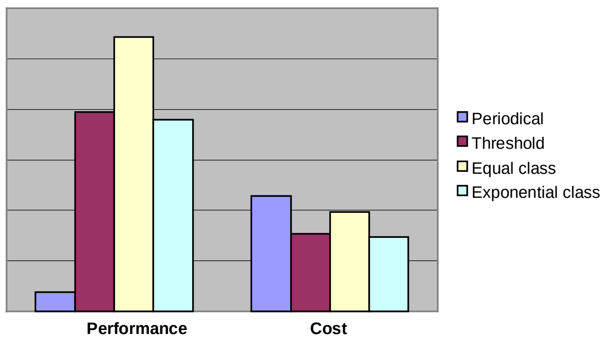

Figure 3 : Matrix with 9 nodes

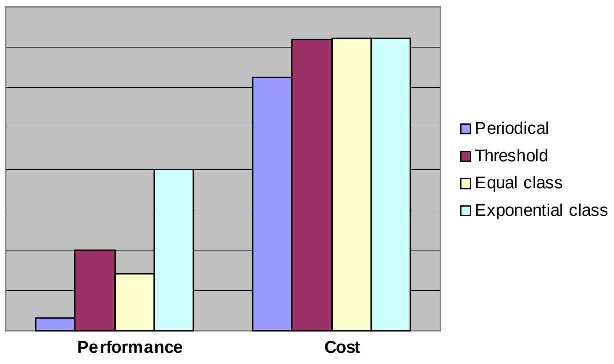

Figure 4 : Matrix with 25 nodes

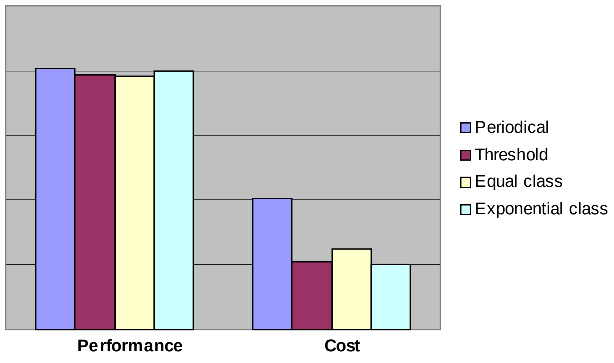

Figure 5 : Tree with 9 nodes

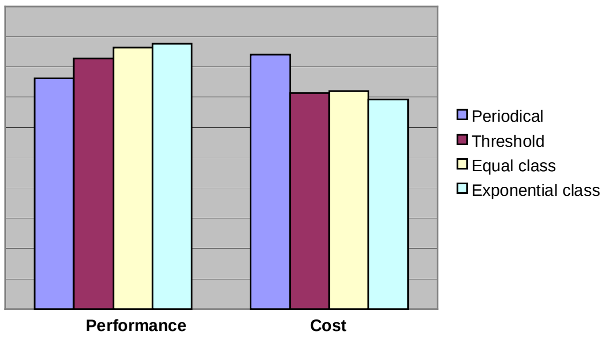

Figure 6 : Tree with 16 nodes

The periodical algorithm performs extremely badly in matrix topologies. Its cost is comparable to the other algorithms, but its performance is much lower. In tree topologies the periodical algorithm can reach a similar performance to the rest of algorithms, but the processing cost is higher. Therefore, the use of alternative algorithms helps to reduce the cost of QoS routing. Thus, with the same cost alternative algorithms achieve a better performance.

The three alternative algorithms behave in a very similar manner over tree topologies. However, the exponential class based algorithm performs slightly better in both large and small trees. Therefore, the gain of using one algorithm instead of another in tree topologies is almost negligible.

These simulations have been done using the same parameters for the algorithms. These parameters can be also modified in the simulator in order to study their influence. In the large matrix the algorithm performing the best was the exponential classed based. However, if the class size is multiplied by ten, the equal class based performs better.

Table 4 : Matrix with 25 nodes [47-48], [49-50]

Algorithm	**Throughput**	**Cost (μs)**
Equal class based	2026 / 1537 packets	35899500
Exponential class based	1759 / 1968 packets	36088500

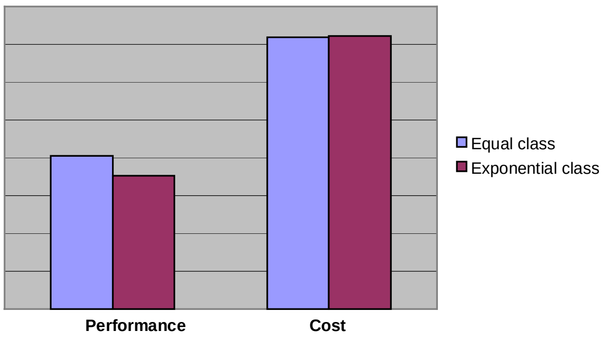

Figure 6 : Matrix with 9 nodes

The following table shows the results of increasing in the same was the class size in the small matrix.

Table 4 : Matrix with 9 nodes [51-52], [53-54]

Algorithm	**Throughput**	**Cost (μs)**
Equal class based	2027 / 652 packets	6451500
Exponential class based	2046 / 633 packets	6264000

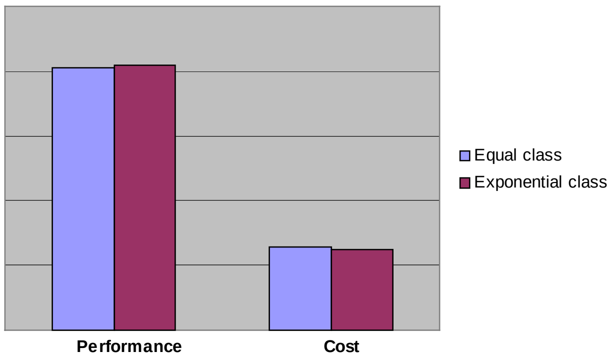

Figure 6 : Matrix with 9 nodes

These results show that exponential class based algorithms behave similar to equal class based algorithms. However, when the available bandwidth in the links varies constantly the exponential class based algorithms can reduce the cost. When the available bandwidth is high the updates are triggered less often and some processing power is saved. In situations where the available is scarce the behavior of both algorithms is practically the same.

5 Conclusions

Triggering link updates in the proper moment helps rise the performance of a system and reduce the QoS routing processing cost. On demand triggers and periodical triggers are far from being optimal methods.

Threshold based updates and equal class based updates behave properly when the available bandwidth level is close to constant and the parameters of the algorithms are adjusted according to that level. If the level of available bandwidth varies a great deal in time exponential class based updates perform better. When the available bandwidth is high the number of updates is reduced. When the available bandwidth is low the updates are triggered more often. This gives this algorithm more flexibility. Exponential class based updates fulfil better the requirements of systems with variable load levels.

6 Acronyms

MaRS: Maryland Routing Simulator
OSPF: Open Shortest Path First
QoS: Quality of Service
QOSPF: Open Shortest Path First with QoS extensions
QRS: QoS based Routing Simulator
RSVP: Resource ReSerVation Protocol

References

[1] Braden R., Clark D., Shenker S., "Integrated Services in the Internet Architecture: an Overview", RFC 1633. June 1994.

[2] Braden R., Zhang L., Berson S., Herzog S., Jamin S., "Resource ReSerVation Protocol (RSVP)", RFC 2205. September 1997.

[3] Apostopoulos G., Guérin R., Kamat S., Orda A., Przygienda T., Williams D., "QoS Routing Mechanisms and OSPF Extensions", RFC 2676. August 1999.

[4] Moy J., "OSPF version 2", RFC 2328. April 1998.

[5] Apostolopoulos G., Tripathi S., "On the Effectiveness of Path-Pre-Computation in reducing the Processing Cost of On-Demand QoS Path Computation", 0-8186-8538-7/98. IEEE 1998.

[6] Apostolopoulos G, Guerin R., Kamat S., Tripathi S., "Quality of Service Based Routing: A Performance Perspective", http://www.tct.hut.fi/~pgzhang/S130/papers/Cost/Apostolopoulos_Perspectives_QoSR.ps

[7] http://www.hut.fi

[8] Zhang P., Kantola R., "Design and Implementation of QRS (QoS_based Routing Simulator)", http://www.tct.hut.fi/~pgzhang/QRS/QRS10/QRS_design.doc

[9] Alaettinoglu C., Dussa-Zieger K., Matta Ibrahim, Shankar A. "MaRS (Maryland Routing Simulator) Version 1.0 User's Manual", http://www.isi.edu/~cengiz/publications

[10] Zhang P., Kantola R., "QRS (QoS_based Routing Simulator) User's manual", http://www.tct.hut.fi/~pgzhang/QRS/QRS10/QRS_manual.doc

[11] http://www.hut.fi/~gonzalo/documents/simulations/matrix3x3.cfg

[12] http://www.hut.fi/~gonzalo/documents/simulations/matrix5x5.cfg

[13] http://www.hut.fi/~gonzalo/documents/simulations/tree9.cfg

[14] http://www.hut.fi/~gonzalo/documents/simulations/tree16.cfg

[15] http://www.hut.fi/~gonzalo/documents/simulations/m9_s_0

[16] http://www.hut.fi/~gonzalo/documents/simulations/m9_s_1

[17] http://www.hut.fi/~gonzalo/documents/simulations/m9_s_2

[18] http://www.hut.fi/~gonzalo/documents/simulations/m9_s_3

[19] http://www.hut.fi/~gonzalo/documents/simulations/m9_l_0

[20] http://www.hut.fi/~gonzalo/documents/simulations/m9_l_1

[21] http://www.hut.fi/~gonzalo/documents/simulations/m9_l_2

[22] http://www.hut.fi/~gonzalo/documents/simulations/m9_l_3

[23] http://www.hut.fi/~gonzalo/documents/simulations/m25_s_0

[24] http://www.hut.fi/~gonzalo/documents/simulations/m25_s_1

[25] http://www.hut.fi/~gonzalo/documents/simulations/m25_s_2

[26] http://www.hut.fi/~gonzalo/documents/simulations/m25_s_3

[27] http://www.hut.fi/~gonzalo/documents/simulations/m25_l_0

[28] http://www.hut.fi/~gonzalo/documents/simulations/m25_l_1

[29] http://www.hut.fi/~gonzalo/documents/simulations/m25_l_2

[30] http://www.hut.fi/~gonzalo/documents/simulations/m25_l_3

[31] http://www.hut.fi/~gonzalo/documents/simulations/t9_s_0

[32] http://www.hut.fi/~gonzalo/documents/simulations/t9_s_1

[33] http://www.hut.fi/~gonzalo/documents/simulations/t9_s_2

[34] http://www.hut.fi/~gonzalo/documents/simulations/t9_s_3

[35] http://www.hut.fi/~gonzalo/documents/simulations/t9_l_0

[36] http://www.hut.fi/~gonzalo/documents/simulations/t9_l_1

[37] http://www.hut.fi/~gonzalo/documents/simulations/t9_l_2

[38] http://www.hut.fi/~gonzalo/documents/simulations/t9_l_3

[39] http://www.hut.fi/~gonzalo/documents/simulations/t16_s_0

[40] http://www.hut.fi/~gonzalo/documents/simulations/t16_s_1

[41] http://www.hut.fi/~gonzalo/documents/simulations/t16_s_2

[42] http://www.hut.fi/~gonzalo/documents/simulations/t16_s_3

[43] http://www.hut.fi/~gonzalo/documents/simulations/t16_l_0

[44] http://www.hut.fi/~gonzalo/documents/simulations/t16_l_1

[45] http://www.hut.fi/~gonzalo/documents/simulations/t16_l_2

[46] http://www.hut.fi/~gonzalo/documents/simulations/t16_l_3

[47] http://www.hut.fi/~gonzalo/documents/simulations/m25_s_2bis

[48] http://www.hut.fi/~gonzalo/documents/simulations/m25_s_3bis

[49] http://www.hut.fi/~gonzalo/documents/simulations/m25_l_2bis

[50] http://www.hut.fi/~gonzalo/documents/simulations/m25_l_3bis

[51] http://www.hut.fi/~gonzalo/documents/simulations/m9_s_2bis

[52] http://www.hut.fi/~gonzalo/documents/simulations/m9_s_3bis

[53] http://www.hut.fi/~gonzalo/documents/simulations/m9_l_2bis

[54] http://www.hut.fi/~gonzalo/documents/simulations/m9_l_2bis