Link State Update Policies for Quality of Service Routing

Bojan Lekovic, Piet Van Mieghem
B.Lekovic@its.tudelft.nl; P.VanMieghem@its.tudelft.nl

Delft University of Technology, Faculty of Information Technology and Systems
P.O. Box 5031, 2600GA Delft, The Netherlands

Abstract

Quality of service (QoS) routing provides increased network utilization compared to the classical best-effort routing. However, effective QoS routing algorithms require frequent distribution of the link state information, which can impose a significant burden on the network resources. In this paper we investigate the triggering link state update policies with respect to the performance measures: link blocking, update error and average time between successive updates. In order to limit the excessive number of link state advertisement messages (LSA) we use two concepts: hold-down timer and moving average filtering. Using simulations, based on the topology independent "local view" simulation model, we compare the triggering link state update policies under both concepts. Our results show that the moving average outperforms the widely accepted hold-down timer concept.

1. Introduction

Quality of service (QoS) routing provides increased network utilization compared to the classical best-effort routing. Furthermore, it fulfils the multiple constraint demands of the communication sessions. Effective QoS routing algorithm requires frequent distribution of the link state information in order to "refresh" the routing tables with most up-to-date information. However, this frequent distribution of the routing information can cause significant burden on the network resources. The mechanism used for efficient generation and advertisement of the link state update information (LSA) is called the link state update policy. The link state update policy can be periodic or trigger-based. In the case of the periodic update policy, each node in the topology periodically advertises information on its available link resources. For the case of the trigger-based policy each node monitors the utilized link resources and triggers the LSA upon a "significant" change. Although periodic updates introduce a predictable overhead for exchanging link-state information, triggered updates can offer more accurate link-state information for the same average rate of update messages.

The impact of the different link state update policies on the QoS routing has attracted relatively small research attention so far. A. Shaikh et al. [1] investigated how the performance and overheads of the QoS routing relate to the link-state update policies. G. Apostolopoulos et al. [2] identified the parameters that determined the QoS cost, namely (a) policy for triggering updates (b) sensitivity of this policy and (c) hold down timers that limit the rate of updates. M. Peyravian and R. Onvural [3] proposed the combined: triggered and periodic update policy to substantially reduce the number of link state updates in the P-NNI framework. However, these studies do not consider other means for limiting the excessive number of link state updates (LSA) than the hold-down timer. The concept of the hold-down timer does not allow for the link state update policy to be adaptive with respect to the changes of the flow arrival rate.

In our paper, we review threshold- and exponential classes-based triggering link-state update policies aimed to provide consistent information on the network resources to the QoS routing algorithm. We perform simulation study in order to compare them in terms of the following performance measures: (1) link state update error, (2) average time between updates and (3) link blocking probability. We further elaborate on problems and propose a new effective technique for limiting the number of LSA – the moving average filtering. This new technique is adaptive to versatile flow rates unlike the hold-down timer. It is based on our observation that the link state trend (filtered link state), rather than every instant link state change, carries relevant routing information.

In our simulation study we show that for the trigger-based link state update policies, proposed moving average technique for limiting the number of LSA is more efficient than the widely adopted hold down timer.

The reminder of the paper is organized as follows. In the section 2 we explain the link state update policies and the metric stabilization problem. Further, to limit the excessive LSA triggering we propose the moving average filter

instead of the hold-down timer. The section 3 describes the evaluation environment and the performance measures we use to evaluate and compare the link state update policies. In the section 4 we summarize our main results and finally give conclusions in the section 5.

2. Link-State Update Policies

Although QoS requirements may be characterized by several dynamic parameters: bandwidth, delay, delay-jitter, and loss, we see the bandwidth as the primary link metric for defining of the QoS. The initial deployments of QoS routing are likely to focus only on bandwidth to reduce algorithmic complexity. Therefore, we choose the link bandwidth utilization as the metric to be monitored in order to decide on link state update. The policies used to send information on the utilized bandwidth can be classified as: periodic and trigger based. In the case of the periodic update policy, the originating router periodically advertises information on the available bandwidth. Trigger based policies monitor the utilized bandwidth and trigger LSA upon a "significant" change. Although periodic updates introduce a predictable overhead for exchanging link-state information, triggered updates can offer more accurate link-state information for the same average rate of update messages. The triggering policies are classified [2] into:

- **Class based triggering policies** (CBT). Link bandwidth is divided into adjacent bandwidth classes and an update is triggered when the current link state value crosses a class boundary. This policy is characterized by two constants: B and f (f >1), which are used to define unequal (exponentially distributed) classes: $(0,B)$; $(B, (f+1)B)$; $((f+1)B, (f^2+f+1)B)$;… For f=1 we get equal classes. With B we set the base class and with f, we influence the number of classes. Very often a large number of updates are generated due to the fluctuation of the bandwidth consumption around the class boundaries. In order to limit the excessive number of unimportant updates, the hold-down timer is used [1,2,3]. The hold-down time is the minimal time spacing between consecutive updates. The hold down timer significantly reduces the number of LSA messages.
- **Threshold based triggering policies** (TBT). An update is triggered when the relative difference between the current and the previously advertised link bandwidth exceeds a certain threshold, e.g. 40%. As the link utilization approaches the link capacity, the advertised values of the available bandwidth are decreasing, thus triggering more frequent updates. Again, excessive LSA messages can be greatly reduced by introducing the hold-down timer.

2.1. The Problem

The concept of the hold-down timer does not allow for the link state update policy to be adaptive with respect to the changes of the flow arrival rate. The hold-down time is a fixed value set by a network administrator and cannot accommodate versatile flow arrival rates. For high arrival rates the changes in utilization are more frequent and the hold-down time is apparently large. Some important changes of the bandwidth utilization cannot be observed during this hold-down time. On the other hand, if the mean arrival rate is small but the flows are long lived and the utilization is still high, the hold-down time will be too short, thus allowing for unnecessary updates. It is difficult to propose a hold-down time that would enable the link state update policy to perform equally well in conditions of versatile flow arrival rates. Finally, sudden bandwidth "spikes" after the hold-down time can be so rapid and so perishable, that the triggering of the LSA could be unnecessary.

2.2. Moving average instead of Hold-down time

The above problem formulation suggests monitoring the bandwidth trend, rather then every instant change in bandwidth, in order to trigger the LSA message. Smoothed bandwidth utilization curve does not carry information on the unimportant changes, which could cause excessive LSAs, but contains valuable information on the utilization trend. In order to accomplish smoothing we "filter-out" the bandwidth utilization curve with the moving average technique. A simple moving average technique computes the mean of the successive values of the bandwidth utilization taken each time upon the arrival or the departure of the flow. The size of an averaged set is called the sample size (N). Each time a newly computed average, based on N past values of real bandwidth utilization, is compared to the previously advertised value. The appropriate relative or absolute change of the "moving" average is monitored to decide on the LSA triggering. The moving average approach is adaptive in nature. If the flow arrival rate is high then the next moving average point will appear sooner and the LSAs will be more frequent. Conversely, the lower the flow arrival rate

(bandwidth change rate) the longer the period until the next moving average point appears and hence, the LSAs are less frequent. The hold-down time is no longer needed.

3. Evaluation Environment

In order to evaluate the performance of the presented link state update policies for both the moving average and the hold-down time concept we use simulations. We choose to simulate the traffic behavior and the link state update policies of one single link (the local view approach), rather than of the whole network. In this way, the simulation time is significantly reduced.

3.1. The Local View Approach

In an arbitrary network we isolate one node and one of its links. We monitor the traffic behavior of this link. We assume that a dynamic routing scheme is deployed. All the traffic source nodes or traversed nodes, that forward traffic to the isolated link, can be logically aggregated into a so-called complex traffic source node. They are the "upstream" nodes of the observed link. The rest of the topology is the collection of "downstream" and idle nodes (nodes that do not influence the link traffic). The complex source node generates aggregated traffic. The mean arrival rate of the traffic generated by the upstream nodes is affected by the change of overall route preference in the following way. The routing protocol advertises periodically, or upon a significant change, the state of the link (e.g. available bandwidth) throughout the network. The link state information is used at each node by the routing algorithm for calculating the new routing tables. The packets or flows are then forwarded according to these new routing tables. As a consequence, the observed link may remain part or be excluded from the shortest paths that previously traversed this link. In our simulations, we modeled the aggregated traffic of the complex source node as a Poisson arrival with the mean arrival rate λ.

The link state information sent back to the network presents the feedback mechanism to improve the network efficiency. After an update, if the link is underutilized, some extra nodes will route traffic towards this link, thus causing an increase in the mean arrival rate λ of the aggregated traffic. Conversely, if the link is over utilized, after the link-state advertisement, fewer source nodes will route traffic via the observed link thereby decreasing the aggregated mean arrival rate. The change in the aggregated mean arrival rate after every update, is modeled by the change-point model:

$$\ldots \rightarrow \lambda_{k-2} \rightarrow \lambda_{k-1} \rightarrow \lambda_k \rightarrow \lambda_{k+1} \rightarrow \ldots$$

As an update occurs at the moment k, the mean arrival rate of the aggregated traffic λ_k updates into λ_{k+1}.

$$\lambda_{k+1} = f(\lambda_k)$$

To reflect the route flapping (oscillation of traffic load) the function f is chosen in such a manner that every next value of the aggregated mean arrival rate is opposite of the previous. A high mean arrival rate is superseded by a low one and vice versa but within certain limits. The λ_{max} corresponds to the full amount of link utilization (link capacity). We limit our observation to the high load area and choose for f the linear function:

$$\lambda_{k+1} = \frac{1}{b-a}(\lambda_k - b\lambda_{\max}) + \lambda_{\max}$$

Different levels of the total average link utilization and the magnitude of traffic oscillations are obtained by changing the slope coefficients: a and b ($a>b>0$).

3.2. Simulation model

In our model we make the following assumptions. Flows arriving to the system are generated according to a Poisson arrival process. There is no queuing of flows. All the flows in the system are served simultaneously. The service time distribution is exponential. There are no elastic flows in the system. Because of the fixed QoS demands (fixed bandwidth), the arrival of a new flow does not cause a departure rate change in other flows. Each flow in the system occupies a certain resource (bandwidth) that we denote as a server. If the resources are fully occupied every new flow is blocked. The queuing model that fits into our assumptions is M/M/N/N [4]. At the output of the M/M/N/N system the bandwidth change is monitored by the link state update policy. The link state update policy decides on when to trigger update. The update value (a sample of the utilized bandwidth) is fed back to the "complex source node" represented by the change-point model. The feedback has a natural delay – flooding delay. Here, for simplicity, we assume that the flooding time is negligibly small. Upon an update, the change-point model calculates the new mean arrival rate. This new arrival rate loads the observed link. In this way we model the route flapping.

4. Results

The performance measures we use to compare different link state update policies are: (a) ***link flow blocking***, (b) ***link state update error*** and (c) ***average time between updates***. We choose the parameters for the link state update policies such that their performance in terms of the link flow blocking and the link state update error is on average alike. In this way, the estimated average time between successive updates can be compared for different update policies.

Simulations were run until 50.000 flows were generated. The first 10.000 calls are considered as warm-up calls and omitted from the statistics. Triggering policies are simulated with the hold down time of the same range and step. In the case of a threshold based update policy, the threshold varies from 10% to 60% with the step 10%. For exponential classes, the base class is set equal to the requested bandwidth per call, while the *f* parameter varies from 1.1 (link capacity is divided into 28 classes) to 2.0 (7 classes) with the step 0.1. Finally, the same policies were simulated with the moving average concept of the metric stabilization (instead of the hold down time), where the sample size varies from 20 to 200 with the step 20. Figure 1 to Figure 6 present the simulation results.

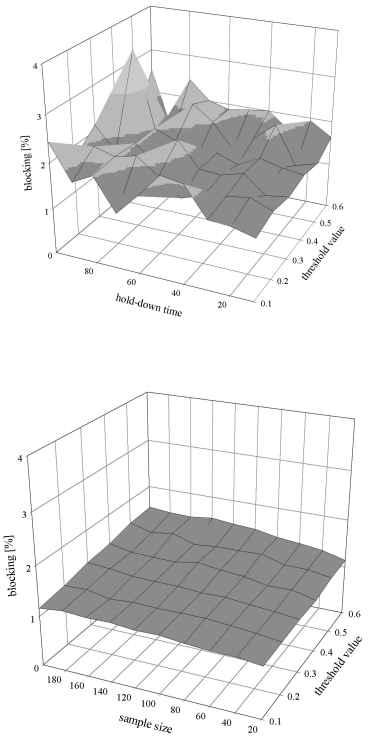

Fig 1. Link blocking-TBT
hold-down time (up), moving average(down)

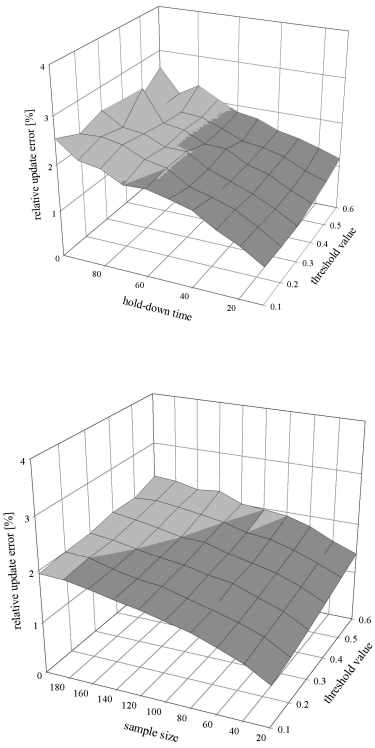

Fig 2. Update error – TBT
hold-down time (up), moving average(down)

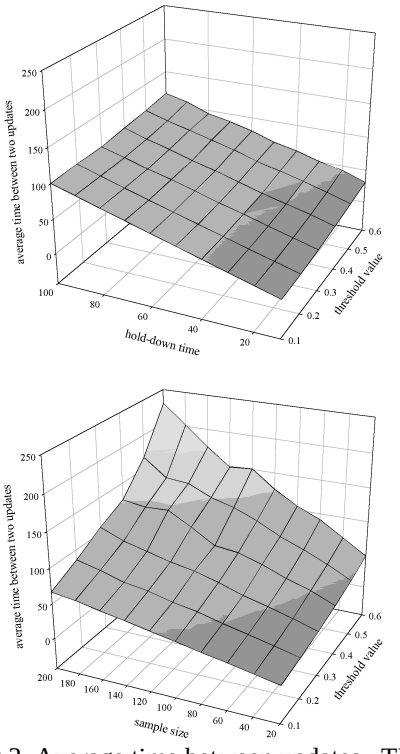

Fig 3. Average time between updates –TBT
hold-down time (up), moving average (down)

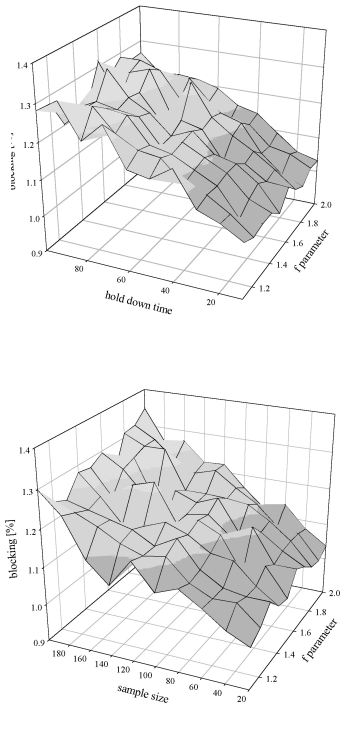

Fig 4. Link flow blocking – CBT
hold-down time(up), moving average(down)

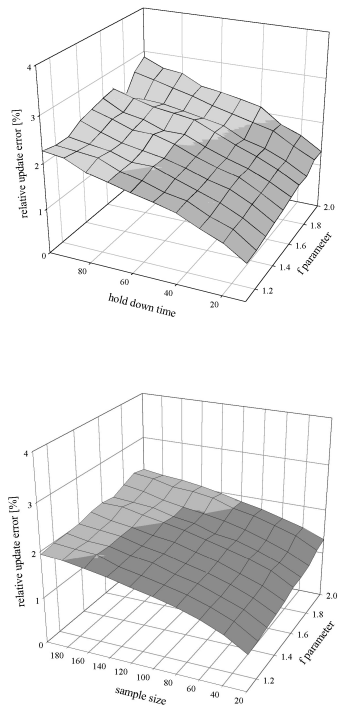

Fig 5. Link state update error –CBT
hold-down time(up), moving average(down)

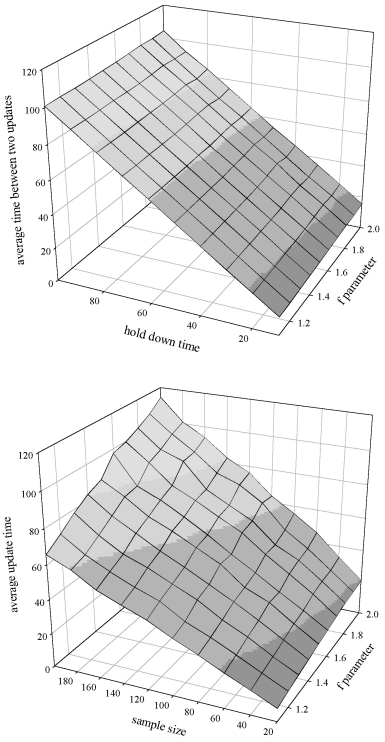

Fig 6. Average time between updates –CBT
hold-down time(up), moving average(down)

Figures 1 and 2 show that the moving average concept of the metric stabilization performs better than the hold-down timer concept hence the blocking and the link state update error are smaller. The figures also show that neither blocking nor update error significantly increase with the increase in the threshold value.

In Figure 3, for the same threshold based policies, the moving average concept significantly outperforms the hold-down timer. For the threshold value set to 60%, the average time between updates, for the moving average is 230 time units and for the hold-down timer 116. For the smaller blocking and update error (Figures 1 and 2), the moving average concept produces 48% less number of LSA messages (and thus less routing protocol and computational overhead).

In Figure 4, the link flow blocking results for the hold-down time and moving average are comparable. The link flow blocking does not increase with the decrease in number of bandwidth classes (increase in *f*).

In Figure 5, the link update error is less in the case of the moving average. Again, for the both, the moving average and the hold-down timer, there is no significant increase in the update error for the decreased number of bandwidth classes (from 28 to

7). These results imply, that the optimal number, for exponential class based policies should be 7.

Figure 6 shows that for 7 bandwidth classes, the moving average outperforms the hold-down timer. The average time between updates is 114 time units, for the moving average, compared to 104 time units for the hold-down time concept. In the same time the link flow blocking is unaffected and the update error is decreased.

The lowest possible protocol overhead and computational cost (the largest average time between updates) is achieved in the case of the exponential class based policy for f=2 (capacity is divided into 7 classes) and in the case of the threshold based policy for the threshold value of 60%. For high loads, the classes are relatively small and the thresholds are easily exceeded.

5. Conclusions

In this paper, we have reviewed periodic and triggering link-state update policies aimed to provide consistent information on the network resources to QoS routing algorithm. We have chosen to monitor the link bandwidth utilization as the most important QoS metric for the link state updates. In order to limit the excessive number of possible updates we have considered the hold-down timer mechanism. Given that fixed hold-down timer cannot adapt to versatile flow arrival rates and may allow for unnecessary updates, we have further considered the moving average filtering of the bandwidth utilization as the mechanism to limit the number of updates. Both update limitation concepts have been evaluated by the simulation study. A topology independent, "local view approach" has been put forward to simulate the single link behaviour under the link state update policies and the conditions of high load. We emulated the network reaction (route flapping) to newly advertised link state by a so-called "change point model".

In the case of the threshold based update policy, the link flow blocking and the link-state update error are fairly stable with the increase in threshold level (from 10% to 60%). The "moving average" approach, for the threshold value 60%, is significantly better than the hold-down timer in terms of the average time between updates (number of LSA messages reduced by 48%). The moving average concept produces smaller update error and smaller blocking.

In the case of exponential classes based update policy, the link flow blocking and the link-state update error are fairly stable with the decrease in number of classes (from 28 to 7). For the optimal number of classes (7) the moving average is better than the hold-down timer. The link blocking and the update error are slightly in favour of the moving average.

Our main conclusion is that the moving average concept reduces excessive number of link state updates in more efficient way than the hold down timer concept proposed in [1,2,3].

References:

[1] A. Shaikh, J. Rexford, and K. Shin, "Evaluating the Impact of Stale Link State on Quality-of-Service Routing", IEEE/ACM Transactions on Networking, Vol. 9, No. 2, April 2001.

[2] G. Apostolopoulos, R. Guerin, and S. Kamat, "Quality of service routing: A performance perspective". In Proceedings of the ACM SIGCOMM '98, pages 17--28, Vancouver, BC, September 1998.

[3] M. Peyravian and R. Onvural, "Algorithm for efficient generation of link-state updates in ATM networks", Computer Networks and ISDN Systems, vol. 29, pp. 237--247, January 1997

[4] Mischa Schwartz, *Telecommunication Networks Protocols, Modeling and Analysis*, Addison-Wesley Publishing Co 1987.