ALCFRA – A robust routing algorithm which can tolerate imprecise network state information

Karol Kowalik and Martin Collier
Research Institute for Networks and Communications Engineering (RINCE),
Dublin City University, Dublin 9, Ireland,
phone: +353 1 700 5805, fax: +353 1 700 5508,
e-mail: {kowalikk,collierm}@eeng.dcu.ie

Abstract

Multimedia traffic and real-time e-commerce applications can experience quality degradation in traditional networks such as the Internet. These problems can be overcome in networks which feature dynamically set up paths with bandwidth and delay guarantees. Multi Protocol Label Switching (MPLS) shows promise as a networking protocol which can provide such capabilities. However, existing routing protocols need to be enhanced or replaced by QoS-aware algorithms if this potential is to be realised.

Several routing methods for selecting paths satisfying QoS requirements have been recently proposed. In practice they must work well in the presence of inaccurate state information.

In this paper we propose the Adaptive Link Cost Function Routing Algorithm (ALCFRA). By modifying the process of generation of link state updates, our algorithm extends the functionality of link state advertisements. The information carried by them represents not only the current link state but also the long term tendency. In the presence of an inaccurate environment this is beneficial and enables the algorithm to tolerate stale state information. We prove the robustness of ALCFRA using simulation results.

1 Introduction

Retrofitting the Internet with QoS capabilities is a challenging task. A major concern is that in the current Internet data packets belonging to the same flow may follow different paths to the destination. Standard best-effort service does not provide any guarantees for the traffic flows and is not suitable for the use by multimedia or real-time e-commerce applications. To provide QoS guarantees new service models [1, 2] and mechanisms [3] need to be implemented.

Protocols such as MPLS [4] supporting explicit routing allow traffic flows to follow a path providing the requested QoS level - as chosen by a QoS routing algorithm. In this paper we consider mechanisms used to find a path with specific bandwidth requirements. Following Kodialam and Lakshman [5] we assume that other constraints such as delay and jitter can be mapped into an effective bandwidth requirement. We assume that the link-state routing algorithm used at each router performs route calculations on an identical network topology. Flooding of link state information is used to ensure that all routers process the same topological and state information.

In this paper we denote each incoming request by its source node s, destination node d and required bandwidth r. Upon receiving a request, a QoS routing algorithm chooses a feasible path. If the entire request sequence and duration of each connection were known, it would be possible to use a multicommodity flow algorithm (MFC) [6], which results in an optimal solution. Typically such information is not provided and this makes MFC algorithms useable only in the phase of off-line optimisation. Most MFC algorithms operate on a graph, with link cost specified as an exponential function of utilisation [6]. Many on-line algorithms, which have to work with connections of unknown duration and without knowledge about future requests, have also followed the idea of using an exponential link cost function [7, 8, 9] or used a cost function with similar shape [10]. The benefit of using of exponential link cost is also intuitively clear: it favours paths on lightly loaded links over those on busy links; so the load is balanced over all links.

In this paper we present the Adaptive Link Cost Function Routing Algorithm (ALCFRA) which works on-line and uses a modified exponential link cost function, adapted to achieve better performance when working with imprecise network state information.

The rest of this paper is structured as follows. In Section 2 various mechanisms for controlling the amount of information flooded by link state advertisements are described. In Section 3 the ALCFRA algorithm is presented. Section 4 presents simulation results showing the benefits of using ALCFRA. Section 5 concludes the article.

2 Link state update mechanisms

The performance of QoS routing methods under various link state update policies has been recently evaluated by a number of researchers [11, 12, 13]. Usually these strategies advertise a state of the link by advertising its utilisation. In this section we propose to represent the state of the link by its cost. The motivation for this approach is the fact that QoS routing algorithms compute routes based on the link cost rather than the link utilisation.

As observed in [11] processing of link state updates is the major source of overhead in QoS routing. Hence, the choice of methods and parameters of mechanisms controlling frequency of link state advertisements should be carefully performed. There are several link state update policies [11, 12, 13], and we have classified them as follows:

timer based policy - uses a timer to control the frequency of link state advertisements. Such an approach allows very precise control of the frequency of updates, but long update intervals are likely to produce fluctuations in link utilisation [13]. This is sometimes called a "magnet phenomenon", because the link advertised as having a low utilisation will probably be favoured by many incoming requests (it will attract new connections) and in the next update interval this will result in a high link load. After advertising a high utilisation few connections will be routed via the link and the whole cycle will repeat.

utilisation change based policy - is used to send link state updates only if the link utilisation changes. Nevertheless the link load can change very frequently. To limit the flooding frequency *hold-down* timers are used to insert a minimal time interval between concurrent updates (from now on we will denote this interval as *hd*). Such update policies can be further divided into:

equal density utilisation change policy - advertises change in the link utilisation with an equal density for high and low loaded links:

- *class based policy* - this partitions the link bandwidth into classes of equal size and every time a boundary between classes is crossed a new link state advertisement is generated.

increasing density utilisation change policy - advertises change in the link utilisation with a density increasing with link utilisation. Such an approach results in less frequent updates for lightly loaded links. This is based on the assumption that a slight increase (decrease) in the link load for a highly utilised link can block (allow acceptance of) the new request. For lightly utilised links this situation happens rarely.

- *threshold based policy* - advertises a new link state whenever the magnitude of the link state change exceeds some predefined threshold. The magnitude of the link state change decreases with the link utilisation. The following notation is used: u_l and b_l are the utilisation and available bandwidth of the link at the time of the last update; u_c and b_c are the current link utilisation and the current available bandwidth; tr is the threshold in percentage (this is the smallest magnitude of change which will be advertised). A link state update will be generated only when:

$$\frac{\mid u_l - u_c \mid}{1 - u_l} * 100 \geq tr,$$

or equivalently when:

$$\frac{\mid b_l - b_c \mid}{b_l} * 100 \geq tr$$

- *class based policy* - this partitions the link bandwidth into classes with size decreasing with increasing link utilisation and every time a boundary between classes is crossed a new link state advertisement is generated.

Algorithms using an exponential link cost function have the property that: a small change in load for low utilised links produces only a small change in link cost, and a small change in load for a highly utilised links can produce a huge link cost change. That makes the use of *increasing density utilisation change* mechanisms appropriate for an algorithms using an exponential link cost function.

Because QoS routing algorithms compute routes based on the link cost rather than the link utilisation we propose to advertise the change in the link cost. This method is simpler and does not generate unnecessary link utilisation updates which do not influence the routing decisions.

It is shown in Figure 1 that *class based equal density cost change mechanism* belongs to the class of *increasing density change mechanisms* for the link utilisation. The approach of advertising link cost instead of link utilisation has the advantage that each node can maintain a different cost function (that feature is exploited in ALCFRA). The main drawback of advertising link cost is that we can not check which links are able to accommodate the new request, and link pruning cannot be employed. However, when dealing with an imprecise state information the process of pruning links that cannot support the new connection is not recommended [13].

3 Adaptive Link Cost Function Routing Algorithm (ALCFRA)

The "magnet phenomenon" described in Section 2 occurs for both *timer based update policies* with a long update interval and for *utilisation change based policies* with a long *hold-down* period; our algorithm aims to decrease the negative influence of the phenomenon.

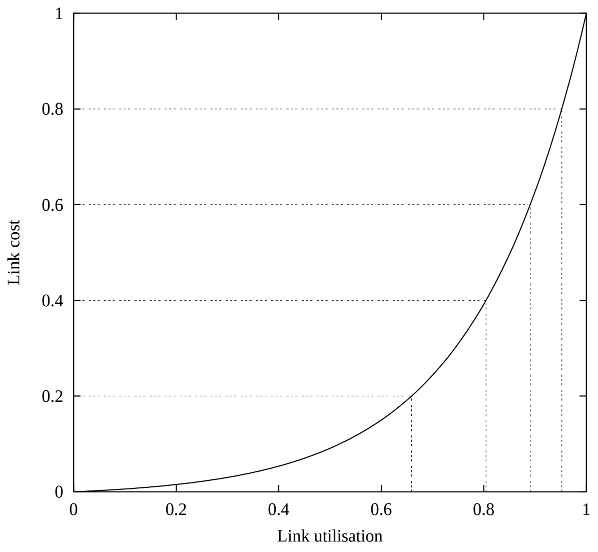

Figure 1: Equal density cost class based update mechanism for exponential link cost function

ALCFRA operates on a weighted graph model *G=(V,E)* (each node $n \in V$ and each link $e \in E$). Let's assume that each link *e* has assigned capacity $c(e)$. Each connection request $\beta_i = (s_i, d_i, r_i)$ specifies the source node s_i, the destination node d_i and the required bandwidth r_i. ALCFRA upon receiving the request β_i computes a least cost path P_i and – if a reservation requiring bandwidth r_i can be made along that path – the request is accepted and the path is marked as P_i^A for the duration of i^{th} connection, otherwise the request is rejected.

The utilisation of the link *e* can be defined as:

$$u_c(e) = \sum_{i, e \in P_i^A} \frac{r_i}{c(e)}$$

ALCFRA as a base cost function of link *e* uses a normalised exponential function with base *A* ([14] presents a method for calculating *A*):

$$cost(e) = \frac{A^{u_c(e)} - 1}{A - 1}$$

But ALCFRA modifies that shape of the link cost function to reflect the long term link state utilisation. In the presence of an inaccurate environment this is beneficial and enables the algorithm to tolerate stale state information. The adaptation of the link cost function aims to prevent situations when a small change in the load for a link which is usually highly utilised would result in a huge change in the link cost. Such adaptation is done using a parameter a_e maintained for each link *e*. Let's assume that the maximal value of a_e is *A* and that a_e can be incremented or decremented by some constant value Δ. The value of parameter a_e fluctuates, tracing the long term link utilisation; it is positive when the long term link utilisation is under the predefined value: u^{tr} – we call this value the threshold utilisation (the meaning of it will be described later), and becomes negative when the link load crosses that value. The parameter a_e changes in the following way:

$$a_e = \begin{cases} a_e + \Delta, & \text{when: } a_e \le A\left[u^{tr} - u_c(e)\right]/u^{tr} \\ a_e - \Delta, & \text{otherwise} \end{cases}$$

The value of a_e is modified as above after every unit of time (in our simulation the unit of time is equal to 1 second). The parameter a_e reflects a longer term tendency in the link utilisation. Finally the link cost function used in ALCFRA is defined as:

$$cost(e) = \begin{cases} \frac{a_e^{u_c(e)} - 1}{a_e - 1}, & \text{if } a_e > 1 \\ u_c(e), & \text{if } -1 \le a_e \le 1 \\ log_{|a_e|}((\mid a_e \mid -1)u_c(e) + 1), & \text{if } a_e < -1 \end{cases}$$

So parameter a_e modulates the link cost function making it more convex when the link load is low (this encourages to balance the load) and making it concave when the link load is heavy.

An example of such an adaptation is shown in Figure 2 for a scenario where $A = 100$ and $u^{tr} = 0.5$. It presents three possible shapes of link cost function: one where $a_e = 60$ and the long term link utilisation is around 0.2 (which is less than u^{tr}); secondly where $a_e = 0$ and the long term link utilisation is around u^{tr}; and thirdly when $a_e = -60$ and the long term link utilisation is around 0.8 (which is more than u^{tr}).

The shape of the link cost function for links with high utilisation (the uppermost curve in Figure 2) is close to the shape of the link cost used in shortest path computation (it uses constant value of 1). So, if the value of a_e is positive and close to *A* it means that link utilisation is low and load balancing is preferred (by the use of exponential link cost function), but if the value of a_e is negative and close to $-A$ the use of shortest-path is recommended (by the use of concave link cost function). This is consistent with the conclusion in [15], that algorithms limiting the hop count result in a better performance for heavy loaded networks, while algorithms balancing the load are beneficial when the load is light.

The threshold utilisation u^{tr} sets the border between the use of concave and convex link cost functions, and its value should be chosen as compromise between good load balancing and limiting the hop count.

The use of such an adaptation together with *equal density cost change update policy* ensures that transient slight changes in the link utilisation will not significantly affect the link cost. This occurs due to the fact that link cost function around the operating point, located at the mean long term link utilisation, is almost flat. So when the link utilisation moves slightly from the operating point (long term link utilisation) link cost remains unchanged. However a significant change in link utilisation will affect the link cost.

Summarising, for each link the cost is calculated as described above and the link state advertisements are generated using an *equal density cost change mechanism*.

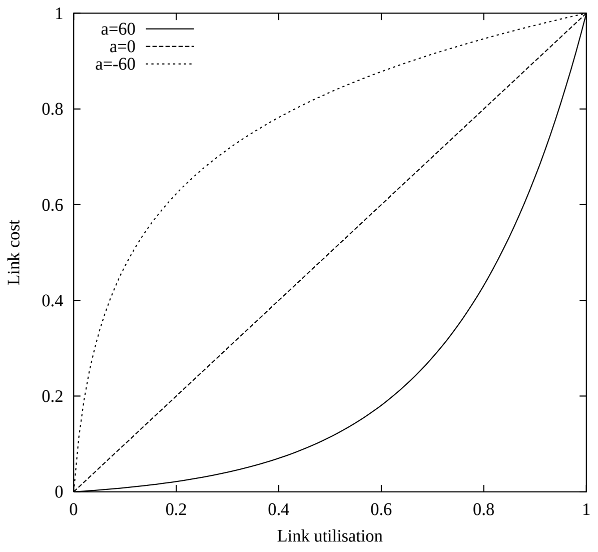

Figure 2: Example of adaptation of link cost function

Upon receiving a route request the least cost path is computed based on the collected link cost information. If the reservation along a chosen path can be realised the request is accepted, otherwise it is rejected.

4 Performance evaluation

4.1 Network Model

The ALCFRA performance was evaluated on the network with the so-called ISP topology [11, 12] shown in Figure 3. In general, the network topology is assumed

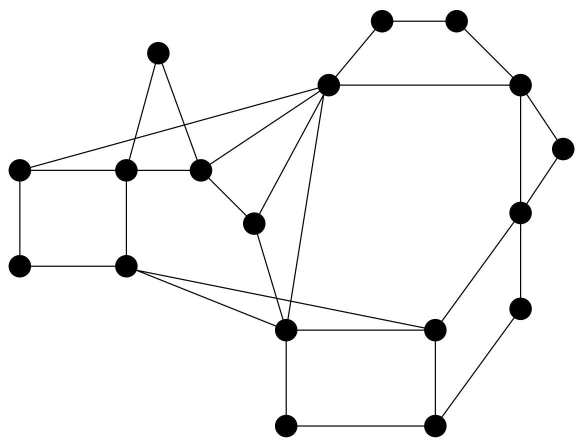

Figure 3: The ISP topology

to consist of N nodes connected using L bidirectional links each with capacity C (for the ISP topology we have used $N = 18$, $L = 30$, $C = 20$). The requests arrive at each node independently according to a Poisson distribution with rate λ and have exponentially distributed holding times with mean value $1/\mu$. The requested amount of bandwidth is uniformly distributed over the interval: $[64kb/s, 6Mb/s]$, with the mean value $B = 3.32Mb/s$. If N^a nodes in the network generate the traffic, it produces the network offered load [12]: $\rho = \lambda N^a Bh'/\mu LC$, where h' is the average shortest path distance between nodes, calculated over all source-destination pairs (for the ISP topology: $h' = 2.36$ if $N^a = 18$).

4.2 Performance metrics

To investigate the performance of ALCFRA we have used the following metrics: *call blocking rate* - defined as:

$$\text{call blocking rate} = \frac{\text{number of rejected requests}}{\text{number of arrived requests}}$$

used to calculate probability of rejecting the new request; *bandwidth blocking rate* - defined as:

$$\text{bandwidth blocking rate} = \frac{\sum \text{bandwidth of rejected requests}}{\sum \text{bandwidth of arrived requests}}$$

which is well suited for calculating blocking probability of requests requiring a different amount of bandwidth [15]; and *update overhead* - defined as:

$$\text{update overhead} = \frac{\text{number of generated link state updates}}{\text{time [seconds]}}$$

used to calculate the overhead generated by the link state advertisements.

4.3 Results

Routing strategies using a convex link cost function increasing with network load such as: *shortest-distance path* [15] - using as cost the inverse available bandwidth, or algorithms using an exponential link cost function [8], or piecewise linear increasing convex function [10], balance the load well over all links. Although these strategies may result in slightly different performance for different network topologies and traffic models, all of them suffer from the effect of the “magnet phenomenon”. Because ALCFRA aims to improve the performance of algorithms using concave function to balance the load, we have decided to compare ALCFRA with the performance of an algorithm using a normalised exponential link cost function (we call it EXP). In all experiments described below we have used a class based equal density cost change update policy with the number of classes equal to 10, and the ALCFRA parameters specified as: $A = 100$, $\Delta = 0.1$, $u^{tr} = 0.5$. The mean connection holding time is 60sec and λ is set to produce required network offered load. We also assumed that each node uses only one hold-down timer with a value *hd* for all links outgoing from the node.

The performance of ALCFRA and EXP under an increasing network load for three different *hd* (hold-down periods): 1sec, 40sec is shown in Figures 4 and 5. When

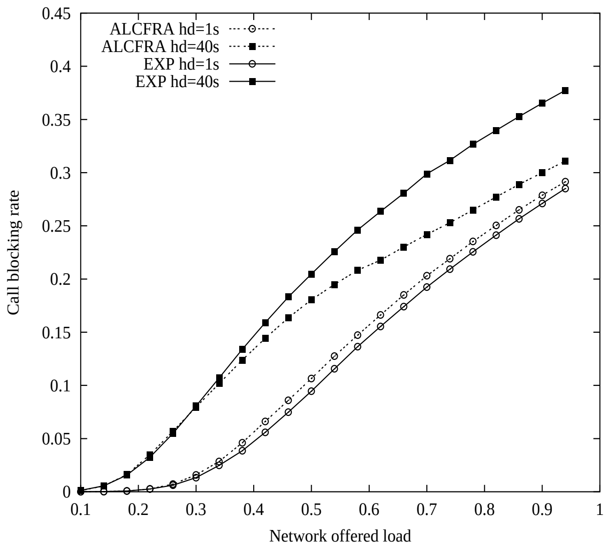

Figure 4: Call blocking probability of ALCFRA and EXP under increasing load

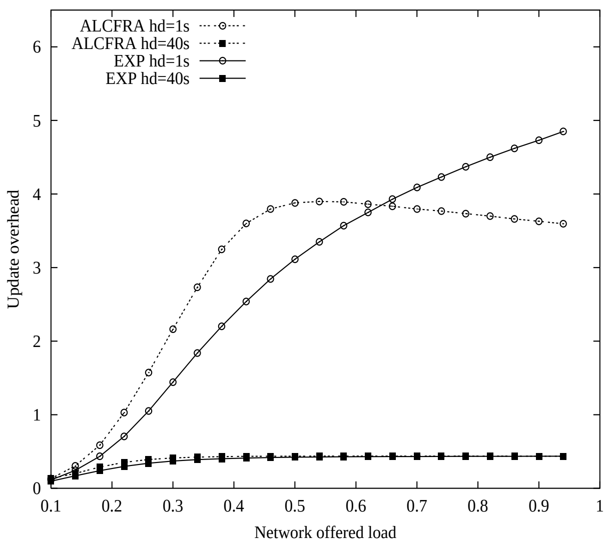

Figure 5: Update overhead of ALCFRA and EXP under increasing load

network load is high and hold-down period is long then the use of ALCFRA seems to be beneficial. Its call blocking probability for small values of *hd* can be compared with EXP, but in a more imprecise environment it gives much better results (see Figure 4). For a small value of hold-down period we can also observe (see Figure 5) that ALCFRA, by preventing traffic fluctuations for usually highly loaded links (load over 0.7 in Figure 5), generates lower update overhead. However ALCFRA adapts slowly to the link utilisation and sometimes the EXP algorithm may need only one link state update to indicate the change in the link load, where ALCFRA in such a situation may generate a few more link state advertisements because of slow modification of the link cost function. If longer hold-down periods are used then both ALCFRA and EXP generate almost the same update overhead.

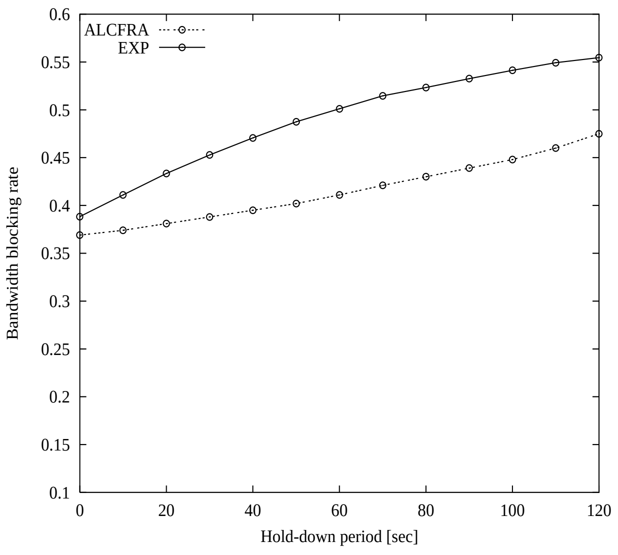

Figure 6: Bandwidth blocking probability of ALCFRA and EXP under increasing hold-down period

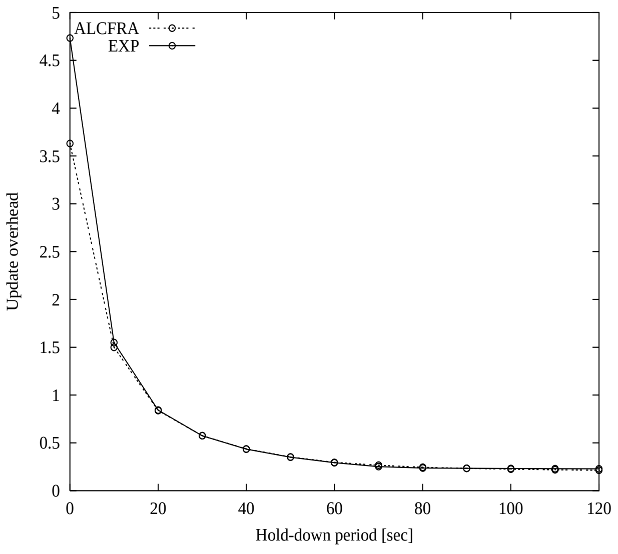

Figure 7: Update overhead of ALCFRA and EXP under increasing hold-down period

Similar results can be observed when the value of hold-down period increases even further, as in Figures 6 and 7. This experiment was realised under offered load $\rho = 0.9$. As is shown in Figure 6, when working with imprecise network information ALCFRA achieves much lower bandwidth blocking probability than EXP. The ALCFRA update overhead controlled by hold-down timers appears to be smaller than or the same as for the EXP algorithm (Figure 7).

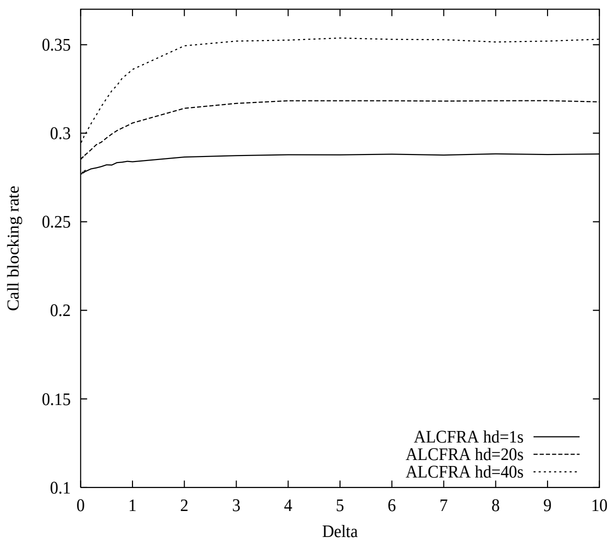

Figure 8: Call blocking rate of ALCFRA for increasing Δ

In Figure 8 we address the issue of the ALCFRA time scale (the experiment was realised under the offered load $\rho = 0.9$). The length of the long-term link utilisation observed by ALCFRA is inversely proportional to the parameter Δ. As shown in Figure 8, longer periods of observation of link utilisation (smaller values of Δ) result in the smallest call blocking rate (hence we have used in other simulations the $\Delta = 0.1$). Results are presented only for a mean connection holding time of 60sec, but if the mean connection holding time increases, we found that long-term link utilisation is more stable and ALCFRA performance improves further. However, blocking probability increases if the mean connection holding time is small (a few seconds). To overcome this we can deal with short connections separately as proposed in [16]. Another solution is to discourage users from requesting bandwidth guarantees for short connections using an appropriate charging scheme.

In Figure 9 we explain why during all simulations we have used $u^{tr} = 0.5$ (the experiment was also realised under the offered load $\rho = 0.9$). When increasing the threshold probability u^{tr} (see Figure 9) the call blocking rate decreases for short hold-down periods, but for longer hold-down periods it grows very fast when $u^{tr} > 0.5$. This fact suggests that when working with actual network state information we can benefit from load balancing approaches (approaches using an exponential link cost function); however if we deal with inaccurate state information, load balancing is appropriate only for lightly utilised

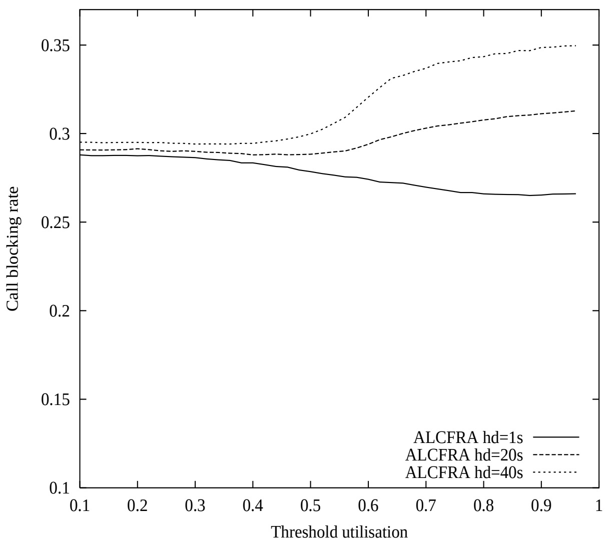

Figure 9: Call blocking rate of ALCFRA for increasing threshold utilisation

links and otherwise the algorithms limiting hop count should be used.

5 Conclusions

The cost of introducing QoS routing mechanisms into the Internet greatly depends on the amount of update traffic [11]. Policies controlling the frequency of link state advertisements introduce inaccuracies and QoS routing algorithms should be able to tolerate such an imprecise environment. Our proposal is to advertise the change in the link cost rather than the change in link utilisation, since QoS routing algorithms compute routes basing on the link cost. In this paper we have presented the Adaptive Link Cost Function Routing Algorithm (ALCFRA) that operates on the advertised link costs. The link cost function used in ALCFRA provides not only information about the current link state - but also gives some insight into the long term link utilisation. This approach is beneficial when the network state is imprecise. Overall, ALCFRA works well even if traffic updates are generated infrequently, and so allows the introduction of QoS routing into the Internet at a reasonable cost.

References

[1] B. Braden, R. Clark, and S. Shenker. Integrated Services in the Internet Architecture: An Overview. *RFC 1633*, July 1994.

[2] D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services. *RFC 2475*, December 1998.

[3] B. Braden, R. Zhang, L. Berson, S. Herzog, and S. Jamin. Resource Reservation Protocol (RSVP) - Version 1 Functional Specification. *RFC 2205*, September 1997.

[4] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. *RFC 3031*, January 2001.

[5] Murali S. Kodialam and T. V. Lakshman. Minimum Interference Routing with Applications to MPLS Traffic Engineering. In *INFOCOM (2)*, pages 884–893, 2000.

[6] S. Plotkin. Competitive Routing of Virtual Circuits in ATM networks. *IEEE Journal on Selected Areas in Communications*, pages 13:1128–1136, August 1995.

[7] B. Awerbuch, Y. Azar, S. Plotkin, and O. Waarts. Throughput-Competitive On-line Routing. *34th Annual Symposium on Foundations of Computer Science*, November 1993.

[8] R. Gawlick, A. Kamath, S. Plotkin, and K. Ramakrishnan. Routing and Admission Control in General Topology Networks. *Technical Report STAN-CS-TR-95-1548*, 1995.

[9] Anil Kamath, Omri Palmon, and Serge A. Plotkin. Routing and admission control in general topology networks with poisson arrivals. *SODA: ACM-SIAM Symposium on Discrete Algorithms*, 1996.

[10] Bernard Fortz and Mikkel Thorup. Internet traffic engineering by optimizing OSPF weights. In *Proceedings of IEEE Infocom 2000*, pages 519–528, March 2000.

[11] G. Apostopoulos, R. Guerin, S. Kamat, A. Orda, and S. K. Tripathi. Intradomain QoS Routing in IP Networks: A Feasibility and Cost/Benefit Analysis. *IEEE Network*, *13(5):42–54*, Sept./Oct. 1999.

[12] Xin Yuan and Wei Zheng. A comparative study of quality of service routing schemes that tolerate imprecise state information. Florida State University Computer Science Department, Technical Report.

[13] Anees Shaikh, Jennifer Rexford, and Kang S. Shin. Evaluating the impact of stale link state on quality-of-service routing. *IEEE/ACM Transactions on Networking*, April 2001.

[14] R. Gawlick, A. Kamath, S. Plotkin, and K. Ramakrishnan. Routing and admission control in general topology networks. *Stanford Technical Report STAN-CS-TR-95-1548*, 1995.

[15] Q. Ma and P. Steenkiste. On path selection for traffic with bandwidth guarantees. *In Proceedings of IEEE International Conference on Network Protocols*, October 1997.

[16] A. Shaikh, J. Rexford, and Kang G. Shin. Load-sensitive routing of long-lived IP flows. In *Proc. ACM SIGCOMM*, pages 215–226, September 1999.