Qualitative Link State Dissemination Control in QoS Routing

Yanxia Jia Ioanis Nikolaidis Pawel Gburzynski
Computing Science Department
University of Alberta
Edmonton, Alberta T6G 2E8, Canada

Abstract *We study the behavior of a Quality of Service (QoS) routing scheme in which connection requests can be assigned to one of k alternative paths from the source to the destination. The multiple–path routing compensates for the inaccurate link state (LS) information. It is found that the frequency of LS dissemination is not the only factor to influence performance. For large LS update periods, the* nature *of the LS information also has a dramatic effect. For example, LS information that inhibits the use of certain links, i.e., links found to be congested when sampled in the last LS update, can block several future requests. We study the problem by considering exceptional additions to the periodic LS updates, when the residual link bandwidth decreases or increases dramatically. We subsequently examine the resulting blocking rate improvement and its corresponding control message overhead. Our findings indicate that certain schemes result consistently in blocking rate improvements for a small amount of additional control overhead.*

Keywords: QoS Routing, Link State Routing, Multiple Path Routing, Flooding

1 Introduction

Link State (LS) routing algorithms have been fairly successful in best–effort network routing. Routers use flooding to disseminate LS information. Each router initiates the flooding of LS information for the links it is attached to. Flooding is a costly operation, with many message exchanges required and the same LS information may arrive to the same node more than once from different paths. Hence, for the sake of scalability, the frequency of flooding updates is generally limited. As a result, the information known to a router about all the links in the network is the result of receiving several LS update message that are, quite possibly, out–of–date and not necessarily globally consistent. That is, they do not represent a consistent global "snapshot" of the network state.

LS routing is a poor match [1, 2] for the requirements of Quality of Service (QoS) routing when the LS information is stale, or otherwise imprecise. The reason is simple. In best–effort routing, the impact of "wrong" routing decisions goes unnoticed, since the users have no performance expectations. In QoS routing, the decisions taken, i.e., whether to admit a particular traffic flow or not, are binary. Connection may, and will (if no path with sufficient resources is found) be rejected. Increased *blocking*[1] is the natural result of out–of–date LS information.

The causes for increased blocking are (a) paths that were known to have enough resources but are now found to have links with insufficient resources, the state of the links has changed and the most recent LS messages received are already obsolete, and, (b) paths that were not considered as usable previously, may now be usable (some of the resources of their links may have been deallocated in the meantime), but without a LS message advertising the fact, the routing algorithm is unable to exploit the change. One expects that as the LS Update Period (LSUP) increases, blocking will increase. We note that what exactly is considered a resource in this context depends on what QoS constraints we wish to satisfy. Generally speaking, we will look into bandwidth as the only resource that is being allocated by the QoS routing algorithms. There are however many more versions of QoS routing for multiple objectives, including delay, delay–jitter, loss probability, bounded number of hops, etc. [3]. We assume that an approach like the one illustrated by Ma and Steenkiste [4] could be, in principle, used to convert (with the aid of particular link scheduling algorithms) performance constraints of delay, delay–jitter and buffer bounds to bandwidth demands, thus avoiding the general case of multi-constraint routing, which is known to be NP–complete [5].

The issue becomes whether it is possible to support QoS routing using stale LS information in a way that would not

[1] We call "blocking" what other authors call *rejection blocking*. That is, a request that fails to be admitted is dropped from consideration. It is neither retried, nor re-negotiated with new parameters.

compromise the efficiency of the network. A classification of schemes addressing this issue can be found in [6] and they are: (a) safety–based routing [7], (b) randomized routing [7], (c) multiple–path routing [3] and (d) localized routing [8]. Safety–based routing treats link state information as "fuzzy" and determines the path with the highest probability of success. The randomized approach calculates a set of feasible paths and randomly selects one. Multiple–path routing probes, in parallel, several paths to determine the best choice. Localized routing simply decides on local information, without the aid of global LS information.

In the comparison carried out in [6], it is found that randomized routing performs the worst. Safety–based routing performance depends crucially on whether routers can correctly infer the "fuzzy" range of LS metric values. That is, it is ineffective, for example, if non-deterministic delays are involved in the propagation of the LS update messages. Localized routing performs well only when the rest of the network LS information is completely worthless by merit of being very obsolete. This is the case for very large LSUP values. The only general–purpose scheme with reasonable performance appears to be multiple–path routing. We therefore provided in earlier studies [9, 10] multiple–path schemes that are simple to implement and do not require any additional LS message overhead, compared to traditional LS routing. We note that [3] advocates a fairly expensive probing scheme because it does not take into account the constraint–reduction approach of Ma and Steenkiste [4]. Instead it deals explicitly with delay constraints and attempts to provide, during the path search phase, a heuristic solution to the relevant NP–complete problem.

Our approach [9, 10] has been that of operating an LS routing algorithm in much the same way that a best–effort network would, that is, to run the routing algorithm only at periodic instants, but instead of determining one path, to determine k paths. In our previous work we have determined that the best selection criterion in terms of blocking is to pick the shortest of the *widest*[2] paths available to the destination. We also determined that for realistic topologies obeying power–law sizes and connectivity, $k = 3$ is generally enough, with larger k leading to diminishing returns in terms of blocking rate improvement. For the calculation of the k shortest paths we use a version of Epstein's k–shortest path algorithm [11].

In the following we revisit the problem of multiple–path QoS routing from the point of view of two properties. First the scope and frequency of LS updates and, second, the qualitative separation of LS update messages into periodic, "generic," and "exceptional" ones. The ideas are rooted on the fact that localized routing holds promise for large LSUP values. In a very large network, LSUP will be large for the sake of efficiency. We would therefore require a compensatory mechanism, in the vein of localized routing, to improve the overall network efficiency.

The remaining sections are as follows: Section 2 presents three schemes for reducing the flooding information without excessively harming the blocking rate of the network. We encounter a rather counter–intuitive behavior which we address in section 3. Based on the observations of section 3, we propose, in section 4, alternative LS update mechanisms to provide better QoS routing at high LSUP values. We conclude with a summary of the results and point out areas of future research.

2 Controlling the LS Information Dissemination

In this section, we will consider the value of inaccurate LS information in a multiple–path routing scheme, with the added twist of trying to incorporate the operating principle behind localized routing. The observation is that, localized routing, as proposed in [8], is extremely restrictive. It is understood that information about links adjacent to a node can be perfect, but should the information of all the remaining links in the network be treated equally and, effectively, ignored? An alternative approach is one whereby the closer a link is to a node, the more important the value of its state becomes. The intuition supporting this approach has to do both with issues of spatial locality (the closer to a node a link is, the more likely it could be used by one or more paths that originate at that node) as well as less inaccuracy (the closer the link is to a node, the relatively faster the link state information will arrive). Based on these observations, we present an alternative that stands between the localized routing of [8] and flooding, while also using the multiple–path approach to QoS routing.

We consider a distance–based update frequency reduction scheme, according to which LS information is sent to the closer neighbors more frequently. Specifically, we consider three variants of the scheme. Deterministic Frequency Reduction (DFR), Probabilistic Frequency Reduction (PFR), and Hop/Distance Threshold (HDT).

DFR and PFR are based on the same idea. Given an update source s, define the h-hop neighbors of s as the nodes h hops away from s. Supposing a node receives m LS update messages from an upstream node, then it forwards only a portion a ($0 \leq a \leq 1$) of the m messages to each of its direct neighbors, rather than forwarding all the m message to them, as flooding would. That is, if there are m update messages generated from s, then each of its h-hop neighbors only receives $m \cdot a^h$ messages, where a is named as the frequency coefficient (FC). The only difference between PFR and DFR is that in DFR, it is deterministically

[2] *Widest* means the paths with the most unallocated, i.e., residual, bandwidth to the destination.

decided how many LS updates are forwarded to the direct neighbors, while in PFR, it is probabilistically determined. For example, if $FC = 0.8$, then with DFR, for every 5 LS updates received, a node forwards only 4 to its direct neighbors. With PFR, however, every time a node receives an LS update, it forwards the update to each of its direct neighbors with a probability of 0.8. Obviously, the extreme case of FC being 1 is actually the case of flooding.

In the third scheme, Hop/Distance Threshold (HDT), there are two update frequency levels. If the distance from a node i to the update source s is smaller than the hop/distance threshold, then node i is updated once every LSUP time, otherwise, i is updated once every two LSUP times. The idea is to update near nodes twice as frequently as far away nodes.

A metric capturing the communication cost savings (in terms of LSUP message exchanges) relative to flooding can be defined as follows:

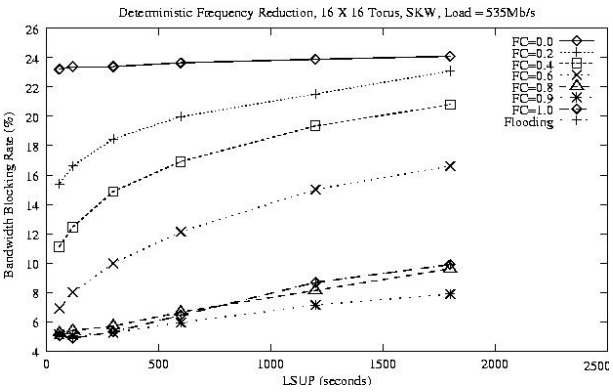
$$\text{Cost Savings} \quad (\%) = \frac{C_{flood} - C_{DFR}}{C_{flood}}$$

where C_{flood} is the cost in the case of flooding, and C_{DFR} is the corresponding communication cost for DFR. Equivalently we define the cost savings of PFR and HDT.

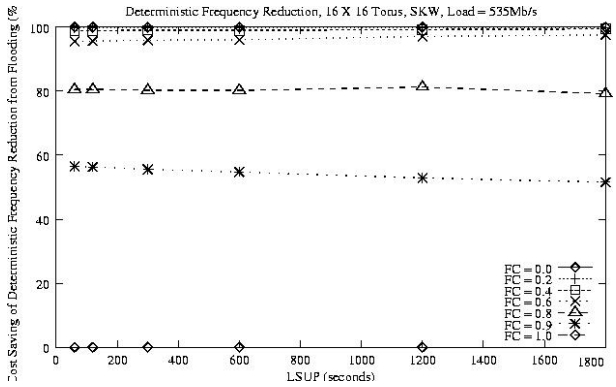
In the following, we investigate impacts of the three schemes on network blocking performance and cost. Since the results for a torus are not representative of real topologies, we simulated both a torus and a realistic power-law based random Internet topology [12]. The torus topology includes 16x16=256 nodes, while the power-law topology includes 250 nodes. All link bandwidths are set to 155 Mb/s, unless otherwise indicated. Connection requests arrive at a rate controlled by the offered load parameter. The requested bandwidth for each connection is generated in a random uniform fashion between the values of 1 and 5 Mb/s. The connection requests follow an exponential interarrival distribution. The connection holding time is lognormal with an average duration of 180 seconds. The multiple–path algorithm uses the Shortest–K–Widest (SKW) selection criterion. That is, for each source–destination pair, the routing algorithm selects from the k widest paths, the one with the least number of hops. If that fails, it continues with the next path, until all k paths have been tried. For the rest of this study $k = 3$.

First let us look into the results of applying a scheme such as DFR on a regular torus topology. Figure 1 illustrates both the impact on the blocking rate as well as the impact on the cost (in terms of messages). Note that the blocking rate is weighted by the connection bandwidth [4, 13], i.e., it is the *bandwidth blocking rate* which is defined as:

$$\theta = \frac{\sum bandwidth\_of\_blocked\_connections}{\sum bandwidth\_of\_requests}$$



(a) Blocking Rate

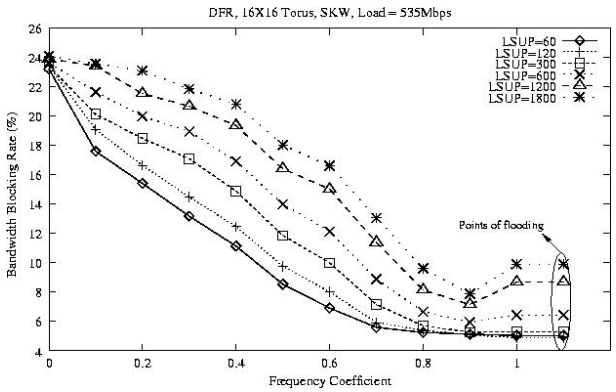


(b) Cost Savings

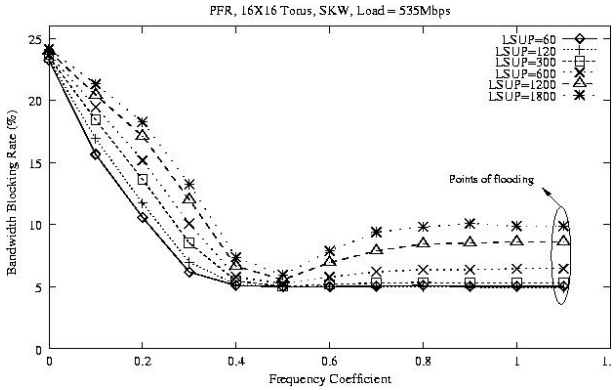
Figure 1: DFR blocking rate and cost savings vs. LSUP period in a 16x16 torus for different frequency coefficients (FC).

The cost reduction for a FC less than 1.0 is dramatic because of the compounding effect from reducing the information as it propagates further away from its source. However, the blocking rate deteriorates only slightly for large values of FC. Indeed the maximum benefits for the least penalty in additional blocking are realizable for an FC of 0.9 in the case of DFR and torus topology.

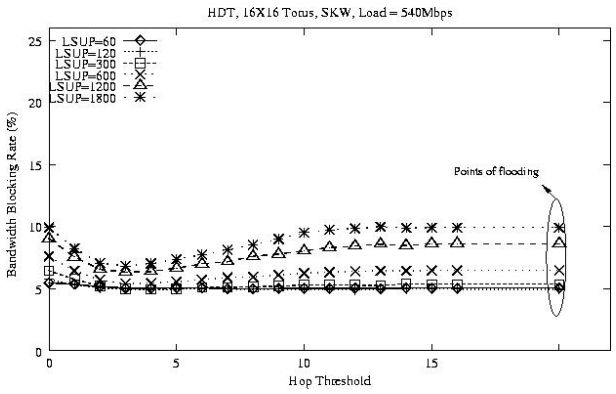
The results of PFR are very similar to those of DFR but the best value for FC is not the same as that for DFR. Figure 2 provides a comparison of the blocking rate results for DFR, PFR and HDT. In all cases there appears to exist a value, either in the FC parameter, or in the hop threshold parameter (for HDT) that minimizes blocking when large values of LSUP are used. That is, for large LSUP, increasing the FC leads to improved blocking up to a point, beyond which any increase leads to a worse, but eventually converging, behavior. Why would more information about



(a) DFR

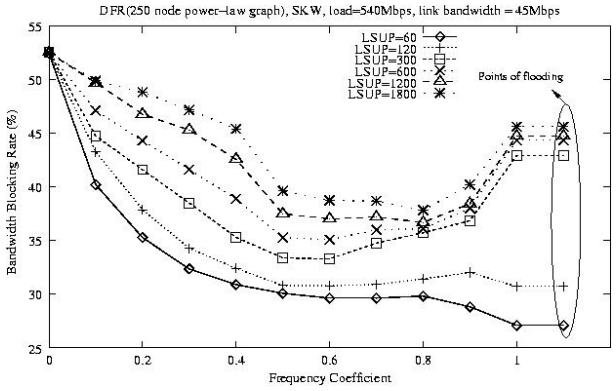


(b) PFR

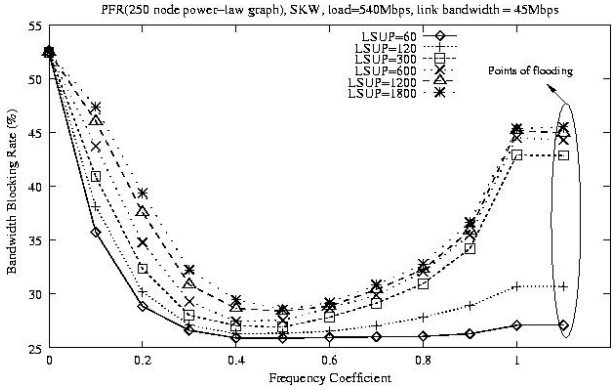


(c) HDT

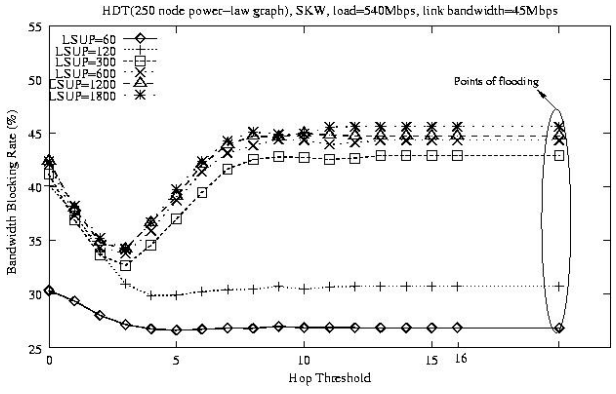
Figure 2: Blocking rate in a 16x16 torus vs. frequency coefficient (for DFR & PFR), vs. hop threshold (HDT) for different LSUP.



(a) DFR



(b) PFR



(c) HDT

Figure 3: Blocking rate in a 250 node power-law topology vs. frequency coefficient(for DFR & PFR), vs. hop threshold (HDT) for different LSUP.

the link state result in worse blocking? For HDT the problem is different but results in a similar behavior. That is, in HDT, the increase of the hop threshold eventually results in a worse blocking behavior. How can it be that *not* knowing about the state of remote links results in *better* choices than knowing about them? These counter-intuitive observations are also true for non-regular, random power–law topologies. Figure 3 presents the results for a random 250 node network following the topology construction in [12]. In fact, the results in realistic topologies suggest an even more dramatic behavior which confirms the same observations we pointed out for the torus topology simulations.

3 Qualitative Impact of LS Information

Why is LS information that is more up–to–date and reaches even further, in terms of the topology, resulting in a worse blocking rate? The phenomenon is particular to large LSUP periods. That is, if LSUP is set to larger values, fresher information may cause worse blocking performance. Since the simulation runs are based on the SKW selection algorithm, we will consider link state information that conveys (residual) bandwidth information. For link l, denote the available bandwidth of this link at the time of last update by B. Some time after the last update, we receive a connection request specifying b as the requested bandwidth. The actual bandwidth of link l at that time, which we do not know exactly, is B'. Essentially, there are four possibilities:

1. $B < b$ and $B' < b$.
2. $B > b$ and $B' > b$.
3. $B < b$ and $B' > b$.
4. $B > b$ and $B' < b$.

Cases 1 and 2 are less interesting because they do not influence the correctness of the routing decision for the current request. Let us focus on cases 3 and 4 that quite drastically change our options regarding the link. In case 3, l is considered infeasible, while in fact it is feasible, whereas in case 4 it is the other way around. The question is now this: what are the consequences of scenarios 3 and 4? We think that the negative impact of case 3 is more significant than that of case 4, because case 3, by underestimating the link capacity, disables a successful route, while case 4, by overestimating the link capacity, just enables an unsuccessful route. Specifically, if at some time t you find a link heavily loaded, i.e., infeasible for a typical request, then its most likely evolution in time (as the LS information becomes outdated) is to become less loaded. It is incorrect to consider that the link remains in the same state for the entire LSUP. Besides, a link that appeared heavily loaded at the time of last update, is quite likely to have been less loaded at some other time, e.g., at some previous update time, which means that relying on the older state we might better approximate its actual current state. Or, to put it differently, if a link was heavily loaded when its last LS was sent, then, in the near future, it will be probably lightly loaded by merit of the fact that all nodes who see the LS information avoid the link in the construction of their paths. As connections on the congested link start terminating, the released bandwidth is not utilized before the next LS packet is sent, which is *not* in the near future if the LSUP value is fairly large.

We conjecture that it makes sense to propagate "good news" faster, by which we mean besides sending out LS updates periodically, to propagate an update whenever a heavily loaded link becomes less loaded. In this way, the inaccuracy of the status information obtained from last update that inhibited the use of a link is quickly corrected.

To examine the accuracy of this claim, we consider a simple 4-node and 4-link toy model, in which four information propagation schemes are deployed, dubbed *Good News* (GN), *Bad News* (BN), *Both News* (GBN), and *Periodic LS Only* (PLS). In GN, aside from periodic LS updates, there are also updates sent whenever the load *decrease* on a link exceeds a certain threshold (in percentage of the link capacity) since the last value reported in LS messages. BN sends updates periodically as well as when the link load *increase* exceeds the threshold. By the same token, GBN conducts updates not only periodically but also when the *increase* or *decrease* of the load of a link exceeds a certain threshold. PLS is the regular periodic update scheme. The simulation result confirms our intuition that "good news" should travel "faster". In other words, it is more important to learn when a loaded link becomes free than when a free link becomes loaded. After all, we are bound to discover the latter anyway if we attempt to setup a connection over it. Further experiments in a realistic environment will be discussed in section 4.

If our explanation about bandwidth based algorithms is correct, then for the hop based algorithms, we will not be able to observe a similar phenomenon, because for this type of algorithm, the major metric, the "shortness" of a path, is a topological property, independent of link bandwidth dynamics. Consequently, in the hop based algorithms, obsolescence of bandwidth information does not influence the routing decision as much as in the bandwidth based algorithms. For the sake of demonstration consider Figure 4, where we illustrate the relationship between the blocking rate and hop threshold for the Widest–K–Shortest algorithm and the HDT scheme. Clearly, the WKS algorithm is insensitive to the LSUP as well as the hop threshold value, confirming our hypothesis.

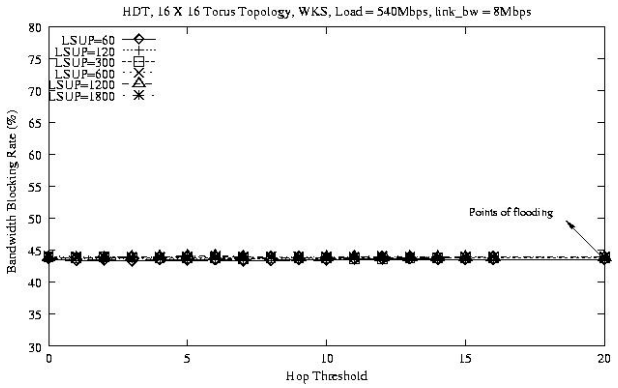


Figure 4: Blocking rate vs. hop threshold for HDT with WKS path selection and different LSUP.

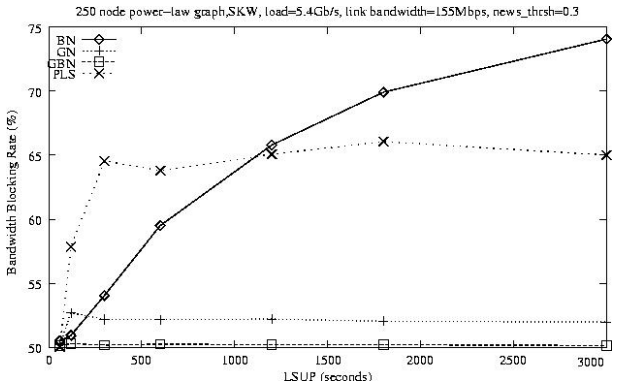


Figure 5: Blocking rate vs. LSUP period on a power–law random topology of 250 nodes.

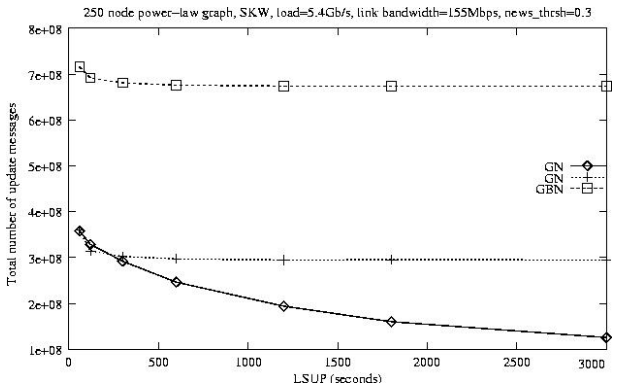


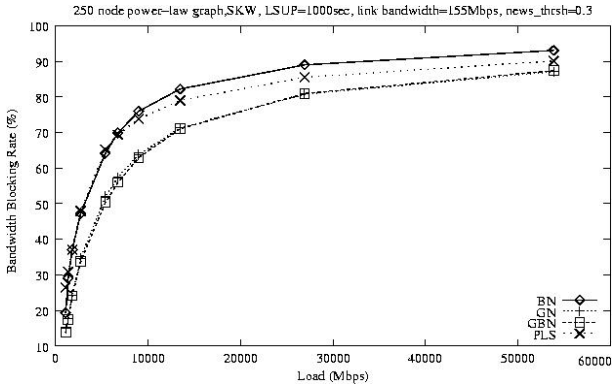
Figure 6: Cost vs. LSUP period on a power–law random topology of 250 nodes.

Figures 2 and 3 also suggests that there exists an upper limit for LSUP. If we exceed that limit, we are better off making routing decisions at random, rather than relying on outdated information. To verify this, we use the aforementioned simple model again and make a comparison among three routing schemes: a perfect knowledge scheme, meaning that each routing decision is based on the perfect knowledge of the network state, periodic LS (PLS) update scheme, meaning updating LS information periodically and making decisions based on the results of the update, and random scheme, meaning picking up a path randomly without depending on the LS information. As expected, the perfect knowledge scheme produces the best performance. For small LSUP, the LS update scheme performs better than the random scheme, while when LSUP is large, the random scheme outperforms the LS update scheme. The result implies that for a periodic update scheme, we should limit the LSUP value. If that value is exceeded, it is wise to make random routing decisions instead of depending on the outdated link state information.

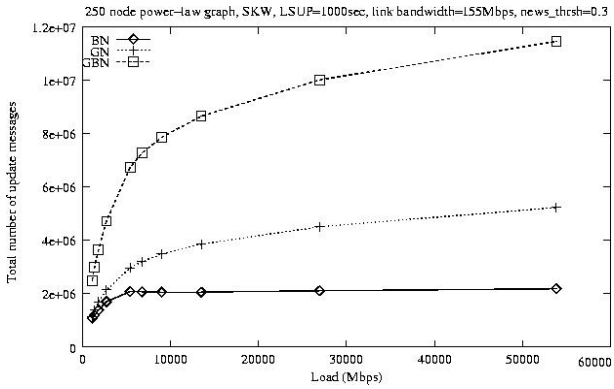
4 Qualitative LS Dissemination Schemes

We will call *qualitative* LSUP dissemination scheme, any scheme for which decisions to send new LS information is triggered on the basis of a drastic quantitative change, sufficiently large to consider that the corresponding link has changed essentially in a qualitative sense. The qualitative changes monitored here are drastic increases and drastic decreases of the residual bandwidth. The notion of "drastic" is captured in the form of a relative threshold. In our study, any change of 30% of the link's bandwidth relative to the previously disseminated LS information of the same link is considered "drastic". From the qualitative point, we classify the schemes as allowing the exceptional propagation of LS information for "drastic" increases (GN), "drastic" decreases (BN), or any "drastic" change (GBN).

We will review the performance of the GN, BN and GBN schemes as they apply to QoS routing using the multiple–path technique to compensate for inaccuracy. The results presented here are for a power–law topology. Similar results are obtained for regular topologies, such as the torus. In all cases, we keep track of the cost introduced by the exceptional messages necessary to convey the "bad" (or "good") news.

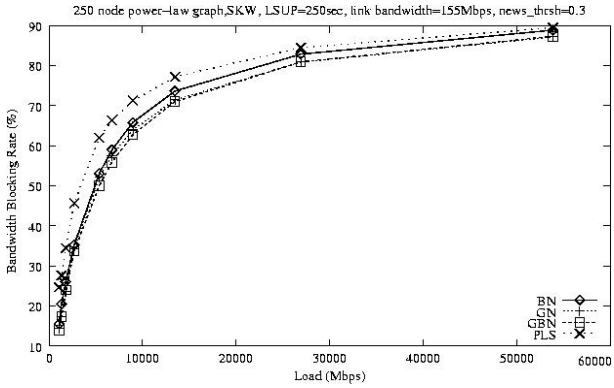


(a) Blocking Rate

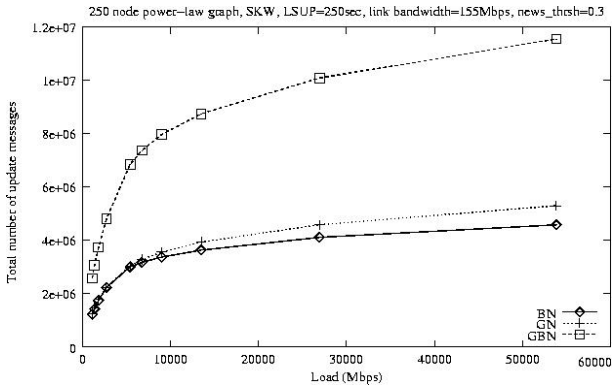


(b) Blocking Rate

Figure 7: Blocking rate and message cost vs. network load for LSUP of 1000 seconds. Power–law random topology of 250 nodes.



(a) Blocking Rate



(b) Blocking Rate

Figure 8: Blocking rate and message cost vs. network load for LSUP of 250 seconds. Power–law random topology of 250 nodes.

Figure 5 reveals that when LSUP is less than a certain point, the traditional PLS scheme performs the worst compared with the other three. This is understandable because in case of a small LSUP, the other three schemes produce more up-to-date information in addition to periodic updates. Among the remaining three, GBN appears to perform the best, BN the worst, and GN is in-between, but the upside for GN is that it achieves a comparable blocking ratio to that of GBN with significantly less message overhead, as shown on Figure 6 and Figure 5.

The second observation is that the BN approach is definitely the worst for increasing LSUP. The explanation is fairly straightforward. Between LS updates, there exist k paths that can be used. If "bad news" arrives then chances are that we are going to restrict the number of feasible paths to something less than k. That is, we start with limited options that, along the way, are limited further, until the next LSUP arrives. This is also the reason that BN turns out to be worse even than PLS.

We conclude by looking into the load–dependent performance of BN, GN, GBN and PLS. In Figure 7 and Figure 8, we vary the load and observe the blocking rate and the number of LS messages sent. It is clear that regardless of whether the LSUP is short or long, the two schemes GN and GBN perform the best. BN's impact is not as detrimental, if the LSUP is sufficiently short, but for a comparable cost in terms of LS messages, GN guarantees a better behavior. For a significantly higher cost one can get the advantage of GBN. However, as Figure 7 and Figure 8 illustrate, the overall improvement cannot be considered spectacular. Nontheless, smaller overhead for GN suggests that it is a reasonable compromise between the blocking rate improvement and the additional message overhead involved.

5 Conclusions

We started by considering multiple–path routing schemes. Based on previous work on k–path selection criteria we attempted to reduce the flooding information. Along with the limited success of such techniques, we encountered a fundamental problem with long LSUP values that, in essence, are useless for QoS routing. We re-considered the idea of what LS information is valuable for the QoS routing algorithms and tailored the LS dissemination scheme accordingly. The final results appear interesting but not quite spectacular.

Confronted by the dilemma of whether it is better to possess state information about the network, even if it is somewhat obsolete, or not to possess any altogether, we note that possessing obsolete state information may inhibit the establishment of connections that would be possible to establish if we were more "agnostic" about the state of the network. However, one has to consider the control overhead of LS information on one hand and, on the other, several connection setup requests that fail due to the lack of knowledge about the state of the network.

We believe that most promising is the direction that associates LS information with an age field. Early LS information can be accepted at the face value. Older information can be considered to be moving to an antithetical direction of the current one (if the residual bandwidth is low now, then we can assume that it will be higher later, etc.). Finally, after a point, random choice may be the best approach. We also note that the studies included here are using randomly uniform selected source–destination pairs for the connection endpoints. The extent of the impact that a biased source–destination selection brings on the performance of QoS routing with controlled LS dissemination will be studied in another paper.

References

[1] G. Apostolopoulos, R. Guerin, S. Kamat and S. Tripathi, "Quality of Service Based Routing: A Performance Perspective," in *Proc. of ACM SIGCOMM*, pp. 17–28, Vancouver, BC, Canada, September 1998.

[2] A. Shaikh, J. Rexford, and K.G. Shin, "Evaluating the Overheads of Source–Directed Quality–of–Service Routing," in *Proc. of ICNP '98*, pp. 42–51, Austin, TX, October 1998.

[3] S. Chen and K. Nahrstedt, "An Overview of QoS Routing for the Next Generation High-speed Networks: Problems and Solutions," *IEEE Network*, pp. 64–79, November/December 1998.

[4] Q. Ma and P. Steenkiste, "Quality-of-Service Routing for Traffic with Performance Guarantees," in *Proc. 5th IWQoS*, pp. 115–126, New York, NY, May 1997.

[5] Z. Wang and J. Crowcroft, "QoS Routing for Supporting Resource Reservation," *IEEE JSAC*, vol. 14, no. 7, pp. 1228–1234, September 1996.

[6] X. Yuan and W. Zheng, "A Comparative Study of Quality of Service Routing Schemes that Tolerate Imprecise State Information," Florida State University, Department of Computer Science, Technical Report TR–010704, 2001.

[7] G. Apostolopoulos, R. Guerin, S. Kamat and S. Tripathi, "Improving QoS Routing Performance Under Inaccurate Link State Information," in *Proc. 16th ITC*, Edinburgh, UK, June 1999.

[8] S. Nelakuditi, Z.–L. Zhang, and R.P. Tsang, "Adaptive Proportional Routing: A Localized QoS Routing Approach," in *Proc. of INFOCOM 2000*, pp. 1566–1575, Tel Aviv, Israel, April 2000.

[9] Y. Jia, I. Nikolaidis, and P. Gburzynski, "Multiple Path Routing in Networks with Inaccurate Link State Information," in *Proc. ICC 2001*, pp. 2583–2587, Helsinki, Finland, June 2001.

[10] Y. Jia, I. Nikolaidis, and P. Gburzynski, "Alternative Paths vs. Inaccurate Link State Information in Realistic Network Topologies," to appear in *Proc. SPECTS 2002*, San Diego, California, July 2002.

[11] D. Eppstein, "Finding the K Shortest Paths," *SIAM J. Computing*, vol. 28, no. 2, pp. 652–673, 1998.

[12] D. Magoni and J.-J. Pansiot, "Comparative Study of Internet-like Topology Generators," Technical Report, LSIIT laboratory, Universite Louis Pasteur, May 2001.

[13] Q. Ma, "Quality-of Service Routing in Integrated Services Networks," PhD thesis, Carnegie Mellon University, January 1998.