Alternative Paths vs. Inaccurate Link State Information in Realistic Network Topologies

Yanxia Jia, Ioanis Nikolaidis, Pawel Gburzynski
Department of Computing Science
University of Alberta
Edmonton, Alberta, CANADA T6G 2E8
{yanxia,yannis,pawel}@cs.ualberta.ca

Keywords: Routing, Resource Allocation, Multiple Path Routing, Link State, Load Balance.

Abstract

We propose a routing scheme in which connection requests with specific bandwidth demands can be assigned to one of several alternative paths connecting the source to the destination. The primary goal of this multiple-path routing approach is to compensate for the inaccuracy of the knowledge available to routing nodes, caused by the limited frequency of link state information exchanges. We study a collection of K-shortest path routing schemes and investigate their performance under a variety of traffic conditions and network topologies, including regular (torus) and realistic (power–law) topologies. We subsequently demonstrate that K-shortest path routing offers a lower blocking probability in all scenarios and more balanced link utilization than other routing methods discussed in the literature. With our proposed approach, it is possible to reduce the frequency of link state exchanges, and the incurred bandwidth overhead, without compromising the overall performance of the network.

INTRODUCTION

With the growing diversification of networking applications and proliferation of integrated services, networks are being forced to cater to a variety of traffic classes with definite and often critical quality of service (QoS) requirements. Traditionally, routing algorithms have ignored the QoS issues and focused on establishing a "reasonable" path between source and destination, typically using the number of hops as the sole measure of quality. In networks predominantly assuming a best-effort delivery paradigm, QoS requirements have been viewed as secondary to routing, with their fulfillment left to other components of the protocol suite, e.g., belonging to transport (call admission) or application layer and thus completely unaware of the routing issues. Nowadays, it becomes more and more evident that the QoS requirements of modern networking applications can no longer be ignored by routing algorithms [7], with their objective being redefined as selecting the optimum path between source and destination that satisfies certain (possibly multiple) QoS constrains. The application of constraint-based routing is seen as central to providing QoS in internetworks, such as the current Internet. Proposed solutions include modifications to the Shortest Path First (SPF) scheme by adding additional constraints that an edge (link) must satisfy in order to be included in the constructed shortest path. Such an approach assumes the existence of reasonable link metrics for the constraints at hand (e.g., per-link residual bandwidth information for bandwidth constraints).

Notably, constraint-based routing calls for explicit/source routing to be used as part of the overall scheme. For example, for the same source and destination nodes, different paths may have to be used by different flows at the same point in time. Such routing flexibility is not available with traditional IP-based routing, which operates based on the destination address alone and, consequently, makes no distinction between routing and forwarding.

A decoupling of these two functions has been proposed recently and implemented using Multiprotocol Label Switching (MPLS) [9]. MPLS provides exactly the flexibility necessary to separate the routing decisions (which can subsequently be based on complicated objectives and constraints) from the simple forwarding mechanism (based on "labels" of local significance, similar to virtual circuit routing). Since MPLS does not assume a particular resource reservation scheme, there is a need for a protocol that performs the actual reservation and maps its result into labels. Towards this end, two protocols have been proposed, CR-LDP [18] and RSVP-TE [5], with a subsequent extension by crankback options [14].

In this spirit, we present a routing path calculation and selection scheme that attempts to mitigate the lack of accurate link state information by exploiting multiple paths to the same destination. It assumes the existence of an RSVP-TE signaling and reservation scheme, coupled with crankback capabilities, like those of [14], and with an underlying MPLS infrastructure capable of performing the forwarding actions.

QoS routing turns out to be considerably more complex than unconstrained routing, especially that some combina-

tions of constraints (e.g., multiple additive constraints) result in NP–hard problems [15], [25]. However, as pointed out in [26] and [20], certain QoS metrics (notably, bandwidth, delay and delay jitter) are not independent when specific scheduling policies are used. In particular, with WFQ–like scheduling, the end-to-end delay and delay jitter depend on the requested bandwidth [26]. Guerin, Orda and Williams [12] showed that with WFQ–like scheduling (under some reasonable assumptions about the traffic pattern), end to end delay constraints can be translated into bandwidth requirements. Consequently, several studies, e.g., [24], consider bandwidth as the sole metric in route computation. As stated in [11], this simplifies to some extent the path computation process, so that tractable solutions can be provided in a number of interesting cases. Following this approach, we also use bandwidth as our sole QoS metric.

Another obvious metric that should be taken into account by a routing algorithm is the length of the connection path expressed as the hop count. The number of hops is a good indicator of the total amount of resources used along the path, which is an important factor from the viewpoint of economy, efficiency and global performance of the network. Our studies indicate that regardless of the other criteria, it always makes sense to keep the connection path as short as possible, as long as it fullfills the QoS constraints.

To implement QoS routing, network nodes should be aware of the state of links, possibly located several hops away. This calls for periodic exchange of link state information, which contributes extra traffic to the network. One problem with QoS routing is therefore the trade-off between the accuracy of the link state information and the overhead incurred by exchanging that information. As the overhead must be kept at a negligible fraction of the total network bandwidth, the link information is bound to merely approximate the true state of the links.

The majority of studies on QoS routing have been aimed at producing a single optimum path. With this approach, only a single path is considered between a source-destination pair, even if there exist some alternative, possibly sub-optimal, paths. In case of congestion, the single path approach is likely to aggravate the problem and may even trigger routing oscillations. In contrast, multiple–path routing would allow different sessions between the same source and destination to be assigned to different paths, depending on the dynamic state of their links. Intuitively, this approach will tend to smooth out occasional problems occurring on some paths, including those caused by inaccurate link state information. This was our primary motivation to embark on the present study.

Most proposals for QoS routing are based on versions of Link State (LS) routing, e.g., [24], [4], [1], [3], [2], [19], [25]. From the viewpoint of a single node, the amount of information that must be processed to keep the link state database up to date grows more than linearly with the number of links in the network. Furthermore, LS information is not merely collected, but must be also relayed to other nodes. Consequently, frequent exchange of link state information may be prohibitively costly, especially in wide area networks of a realistic size. On the other hand, infrequent exchange, and stale state of links, may severely impair the quality of routing. Formulated in this context, the problem of LS–based QoS routing is how to get acceptable performance in networks with inaccurate link state information.

Formally, we address in this paper a bandwidth–constrained multiple–path routing problem described as follows: Given a network represented by a Directed Acyclic Graph (DAG) $G(V, A)$, the capacity of each link, b_{ij}, where $ij \in A$, and a bandwidth request B for a connection from node s to t, find the best K paths from s to t $\{P_k | 0 < k \leq K\}$, such that $b(P_k) \geq B$, $\forall 0 < k \leq K$, where $b(P_k) = min_{ij \in P_k} b_{ij}$ is called the bottleneck bandwidth of path P_k. Besides specifying the optimality criteria that define the "best" paths, we have to also describe the construction and selection strategies for the K paths. The path construction is performed each time new LS information becomes available, while the path selection is performed for each connection request.

We develop two categories of K–shortest routing algorithms, hop–based and bandwidth–based, and correspondingly five path selection algorithms: Best–K–Widest, Random–K–Widest, Shortest–K–Widest, Best–K–Shortest, and Widest–K–Shortest. Although these protocols have been studied to some extent in our previous work [16], those experiments involved very small networks and their results cannot be generalized onto realistic wide-area environments. In order to appreciate the impact of the plurality of paths ($K > 1$), in this paper we extend the previous work to significantly larger networks (up to 1000 nodes) and topologies that have been [21] demonstrated to capture the current Internet topology in a more realistic way.

Our results demonstrate that routing based on multiple paths gives better performance in terms of blocking rate and load balance than single-path solutions, if the link state information is inaccurate. We also show that some multiple path solutions may fail to take advantage of their supposedly "bigger choice," if they do not account properly for the possible inaccuracies in link state information. We conclude that hop–based algorithms, i.e., ones using hop count as the primary metrics, tend to outperform bandwidth–based solutions, and that the network topology has a paramount impact on the behavior of a routing algorithm.

RELATED WORK

Recent studies on multiple path routing include [13], [8], [23], [12]. Work somewhat similar to our study has been reported in [12], where *multiple paths with equal cost* are constructed and used at the same time. However, an "equal-cost multi-path" does not necessarily exist for all source-destination pairs, which restricts the applicability of that approach. Another solution has been proposed in [8], [23],

whereby all the multiple paths are used in parallel to route a single traffic stream. In contrast, our scheme is essentially sequential, with every single connection being set up along one specific path. The parallel multi-path routing scheme [8], [23] reduces the reservation delay but it suffers from synchronization problems and requires reassembly buffers at the destination to account for traffic arriving out of order [7]. None of those studies investigates how the performance of the routing algorithms relates to the accuracy of the link state information.

In [6], the authors present a K–shortest paths algorithm and evaluate its performance. However, that work is based on a fully–connected network and is not applicable to realistic wide area networking, e.g., the Internet. The Bellman–Ford based Widest–Shortest algorithm studied in [2] also provides multiple paths between a source-destination pair. However, it ignores the equal–hop–count multiple paths, which property, as we shall see later, impairs the performance of the routing protocol if the link state information is inaccurate. Ma and Steenkiste [20] investigate several routing schemes, including the so-called *Dynamic–Alternative (DA)*, and compare their performance under a variety of topologies and traffic conditions, but again they ignore the issue of accuracy of the link state information. In this paper, we compare our Widest–K–Shortest (WKS) variant to DA, with the link state information being accurate as well as inaccurate.

The problem of inaccurate link state information was investigated to some extent in [11] by evaluating the probability of success (the so-called "safety") of a path consisting of links for which the LS information is inaccurate. In contrast to that approach, we propose to dissipate the possible inaccuracy of the state information concerning a single link by admitting multiple paths between a given source-destination pair.

Several papers discuss the algorithms for finding K shortest paths [10], [22]. Our solutions are based on the algorithm presented in [22], which we have modified to find K best one–to–all loopless paths.

THE ROUTING MODEL

Link state routing requires each router to maintain a link state database, which is essentially a map of the network topology with associated resource allocation. When a network link changes its state (i.e., goes up or down, or its utilization is increased or decreased), the network is flooded with a link state advertisement (LSA) message. After the link state database at each router is updated, the router will re-calculate its routing tables to all destinations. LSA messages can be issued periodically or when the actual link state change exceeds a certain relative or absolute threshold [4]. Obviously, there is a trade-off between the frequency of state updates (the accuracy of the link state database) and the cost of performing those updates, both in terms of the extra network bandwidth and the processing time at the routers.

We assume in our model that after every update, each node immediately has the full knowledge of the current link states in the whole network. This is justifiable because the propagation delay of an LSA message is generally small compared to the length of the update period and to the duration of a traffic session. However, we do capture the fact that the updates do not occur continuously but periodically, with a period equal to the LS update period (LSUP).

A source–destination path can be computed upon request, i.e., when it is demanded by the source, or precomputed in advance, e.g., periodically. In our study, routes are precomputed periodically, with the computation period being equal to LSUP. Thus, by setting LSUP = 0, we can model the behavior of our routing schemes with accurate link state information and with path computation performed on demand.

The path selection process is carried out for every connection request and is separated from the path computation algorithm. For a given destination, the source router is presented with K paths to choose from. The actual ordering of those paths depends on the specific scheme being used and will be detailed in subsequent sections. If the connection cannot be established via one of the paths, e.g., due to the insufficient remaining bottleneck bandwidth, the router picks one of the remaining paths and tries again. The request is blocked if no path is found after all K of them have been attempted.

PATH CONSTRUCTION

Our path construction algorithm [16] is a label–setting algorithm based on the Optimality Principle and being a generalization of Dijkstra's algorithm [22], which we have modified to find K *one–to–all* loopless paths instead of *one–to–one* non-loopless paths. Its space complexity is $O(Km)$, where K is the number of paths and m is the number of edges. By using a pertinent data structure, its time complexity can be kept at the same level $O(Km)$ [22].

With the hop count and path bottleneck bandwidth used as two separate metrics, our algorithm will generate K paths that are either *shortest* in terms of the number of hops (hop–based algorithms), or *widest* in terms of the bottleneck bandwidth (bandwidth–based algorithms). In order to limit the impact of requests for trivially small bandwidth, a threshold is used to prune unsuitable links right away, at the path construction stage.

PATH SELECTION

Below we list five path selection algorithms resulting from two different major criteria and different ways of applying the minor criteria.

- BKW — Best–K–Widest: from the K widest paths, select the one whose bottleneck bandwidth most tightly fits the request bandwidth (best–fit).
- RKW — Random–K–Widest: from the K widest paths, select one at random.

SKW Shortest–K–Widest: from the K widest paths, select the one with the least number of hops.

BKS Best–K–Shortest: from the K shortest paths, select the one whose bandwidth most tightly fits the connection request.

WKS Widest–K–Shortest: from the K shortest paths, select the one with the largest bandwidth.

Note that, although the names *Shortest–K–Widest* and *Widest–K–Shortest* resemble *Shortest–Widest* from [25] and *Widest–Shortest* from [2], our algorithms are quite different from those previously proposed solutions. In particular, our SKW finds the top K paths, while the *Shortest–Widest* algorithm of [25] uses the "shortest-widest" constraint during the (single) path construction phase. That is, when multiple labels with the same bandwidth are found, only the shortest one is permanently labeled and the others are deleted. SKW maintains all the widest labels for later use and is therefore able to find top K paths. It decides on the "shortest" path during the path selection phase rather than when the path is being computed.

Furthermore, note that although the *Widest–Shortest* algorithm from [2] also generates multiple paths for a given destination, it is different from WKS in that it provides fewer choices for selecting short paths, and the resulting connection is thus likely to need more resources. In particular, it uses an upper bound for the hop count and for each hop count, it only keeps one widest path. Consequently, it ignores the equal–hop–count multiple paths. Besides, the *Widest–Shortest* algorithm requires that the $i+1$–th path be "wider" than the i–th path. This makes sense when the link state information is accurate, but if it is not the case, some feasible paths may be incorrectly ignored. Thus, the very nature of the algorithm in [2] impairs its performance when the link state information is inaccurate.

THE SIMULATION MODEL

In our simulation model, the link state information is updated periodically, with LSUP varying from 0 to 100 minutes, to create scenarios with different levels of accuracy.

The most representative of our simulation results have been obtained for random network configurations built by the generator of Magoni and Pansiot [21]. Reasonably large networks generated by this program have been demonstrated to obey the most relevant (from the viewpoint of routing) power laws found in existing wide area networks (e.g., sections of the Internet). A random network configuration in our experiments is characterized by two parameters: the number of nodes N, and the average node degree δ. A typical value of δ found in the Internet is 2.5 [21]. We consider sparser networks, with $\delta = 2.0$, as well as denser networks, with $\delta = 2.9$. For reliability, a single experiment has been verified on a number of different topology samples generated for a given pair $< N,\delta >$. For a network of a reasonably large size (≥ 500nodes), the obtained results are highly consistent across different topology samples (obeying the same power laws).

For reference, we also consider regular torus networks with the same populations of nodes as the power law configurations. Certain performance measures, notably our *coefficient of variation* (see below) may be affected by the inherent irregularities present in any random network, even if it is large and obeys reasonable statistical laws. A regular topology eliminates those problems and helps us find out how much our performance measures are influenced by statistical aberrations in network configurations.

Every link in our network has the same bandwidth of 155Mbps. The offered traffic consists of randomly generated sessions with exponentially distributed interarrival time. The mean value of this distribution is a simulation parameter that determines the global load in the network. The duration of a session is log-normally distributed with a mean of 180 seconds, and its bandwidth is uniformly distributed between 1 and 5 Mbps. The source-destination pair is selected at random: every station in the network is equally likely to be selected for this role.

The total amount of simulated time for a single experiment was originally set at 24 hours, and six independent experiments were used to obtain a single point of a performance curve. The high observed consistency of results allowed us to reduce the amount of simulated time by half, and the number of samples by the same factor.

We are primarily interested in two performance measures as functions of the offered load and LS update period: the connection blocking rate and the *coefficient of variation* of the link utilization. The blocking rate is weighted by the connection bandwidth [19], [20] and defined as:

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

The *coefficient of variation* of the link utilization captures the variability of the load between different links and indicates how well the offered load is spread over the network resources. We define it as:

$$C_v(LU) = \frac{Std(LU)}{u(LU)},$$

where LU stands for link utilization (i.e., the fraction of time the link is being used), *Std* is the Standard Deviation and u is the mean.

THE RESULTS

The goal of our experiments was to answer the following questions:

1. Is there a single path selection algorithm (among the ones listed in Section) that gives consistently the best performance?

2. How does the multiple-path approach fare in the context of inaccurate link state information? In particular, how does

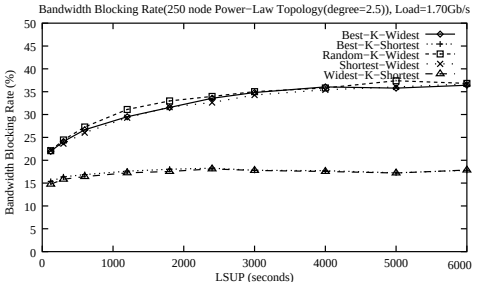

Fig. 1. Small typical network, $k = 3$

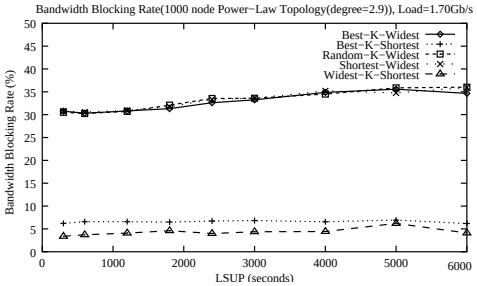

Fig. 2. Large dense network, $k = 3$

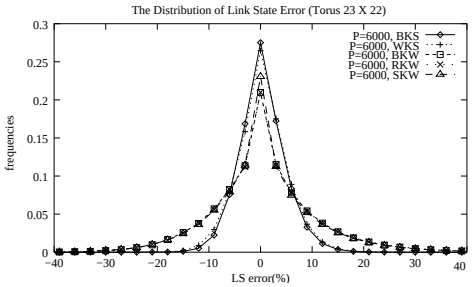

Fig. 3. Link state error distribution

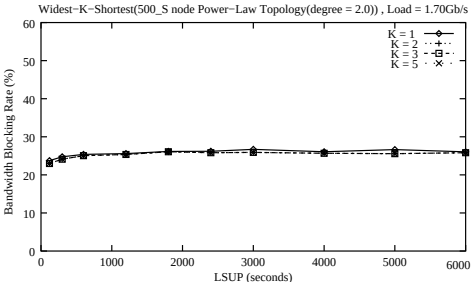

Fig. 4. Medium sparse network, different values of k

the performance of our multiple-path routing scheme depend on the LSUP?

3. How large should the multiple-path selection be, i.e., how does k (the number of paths to choose from) affect the performance of our routing scheme?

4. What are the costs and benefits of the proposed multiple-path algorithms?

5. How does our routing scheme compare to other schemes?

Fig. 1, obtained for a small and typical network, illustrates the blocking rate under all five path selection algorithms with the number of alternative paths $k = 3$. The trend shown in this figure has been clearly visible in all our experiments. First, it turns out that the hop–based selection algorithms (i.e., WKS and BKS) unquestionably outperform the bandwidth–based ones (BKW, RKW and SKW), with WKS winning over BKS. Although the superiority of WKS over BKS is not obvious in Fig. 1, it becomes visible in a larger/denser network, e.g., see Fig. 2.

The reason why the hop–based algorithms win in this competition is that by making the path length the primary optimization criterion they focus on minimizing the amount of resources needed to sustain a connection. Consequently, the network tends to use less of its total bandwidth per session and is thus able to accommodate more connections.

The two figures also demonstrate that past some narrow initial range of LSUP, the quality of our routing schemes (especially the hop–based ones) is not adversely affected by the inaccuracies in the link state information. In all cases, as the LS update period goes to infinity, the blocking rate stabilizes to a constant. This is not surprising, since then the calculated paths are not much better than ones selected at random. A more important observation is the relative insensitivity of the hop–based schemes to the LS update period.

Aside from blocking performance, we also made a comparison between the link state error distribution associated with the two types of algorithms. Based on Fig. 3 and using a quantile-quantile plot method [17], we found that the error is approximately normally distributed. The figure indicates that the hop–based algorithms generate a smaller variance of error. According to the graph, we found that with the hop–based algorithm, BKS, 98.40% errors lie in a range as narrow as $[-10\%, 10\%]$, while with the bandwidth–based algorithm, BKW, only 66.41% of errors are in that range, and the remaining 35.59% are out of this range, meaning that the hop–based algorithm generates smaller LS errors than the bandwidth–based one.

To answer question 2 and 3, we examined the impact of k on the performance of a path selection algorithm, and found it to depend on the algorithm and on the network density. Fortunately, the hop–based schemes perform quite well with

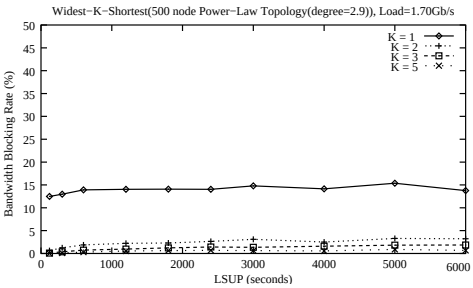

Fig. 5. Medium dense network, different values of k

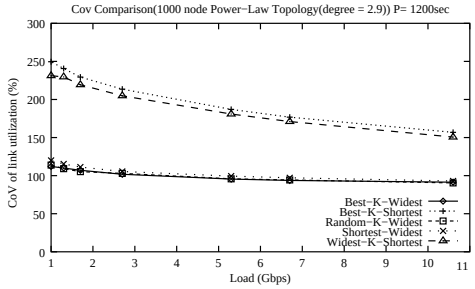

Fig. 6. Large dense random network, $k = 3$

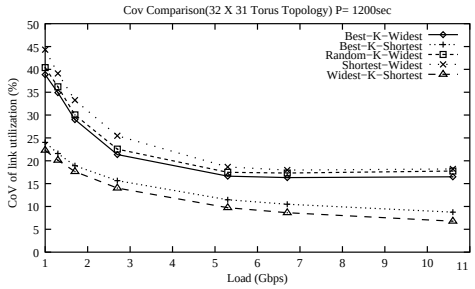

Fig. 7. Large regular network, $k = 3$

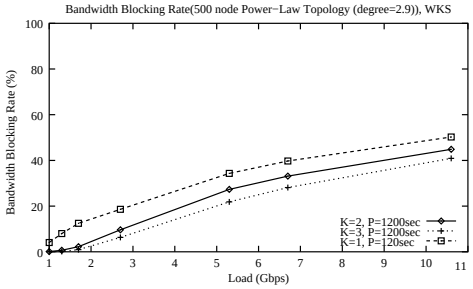

Fig. 8. Blocking rate comparison of multiple path and single path routing

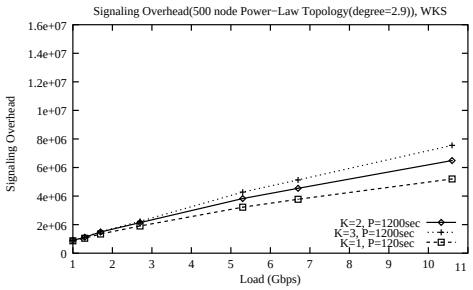

Fig. 9. Signaling overhead comparison of multiple-path and single path routing

a small number of alternative paths to select from. Fig. 4 and 5 show the blocking rate of WKS in two medium-sized networks for different values of k. In the sparse network, the impact of k is quite negligible, which means that alternative path selection brings about little, if any, improvement. This is understandable, because low connectivity implies low number of alternative paths with comparable length (using a comparable amount of network resources). Consequently, even though multiple alternative paths may still exist, they tend to be of different length, so selecting an alternative to the shortest path in such a sparse network is statistically a poorer choice than in a network offering multiple shortest paths. In Fig. 5, we see a clear improvement caused by the statistical presence of sensible alternatives.

Regardless of the circumstances, the gap between the cases $k = 1$ and $k = 2$ is significantly bigger than between $k = 2$ and $k = \infty$. This indicates that while it is advantageous to have a choice, that choice need not be excessively big. This observation is good news. It means that the complexity of the routing algorithm can be well contained, because all it needs to do is to find just a few (e.g., 3) alternative paths, which effort is only moderately bigger than finding a single path.

Our primary intuition behind multiple paths was that with several alternative routes, we would be able to spread the offered load more evenly over the entire network. Thus, one would expect that the better path selection algorithms should result in a lower observed value of the *coefficient of variation* of the link utilization C_v.

Alas, as indicated by Fig. 6, this kind of study must be done on a regular topology to make sense. With local irregularities that are bound to occur in a random network of any size, by trying to consistently follow the shortest paths, a routing scheme may quite legitimately create localized hot spots. Those departures from a uniform spread of the offered load will tend to increase with the decreasing average length of a connection path, because the irregularities in topology manifest themselves on a small scale, and they tend to disappear when we look at larger regions of the network. Consequently, in Fig. 6, the observed value of C_v is higher for the path selection schemes that offer better performance, i.e., target shorter paths. On the other hand, in Fig. 7, obtained for a perfectly regular torus network, the *coefficient of variation* shows a reversed trend, which remains in a perfect agreement with intuition.

So far we have only examined the performance improvement of the proposed multiple-path scheme, in the following we will look at the cost of it. The major overhead in our routing protocol consists of the signaling overhead and the link state exchange overhead. With QoS routing, we need to make

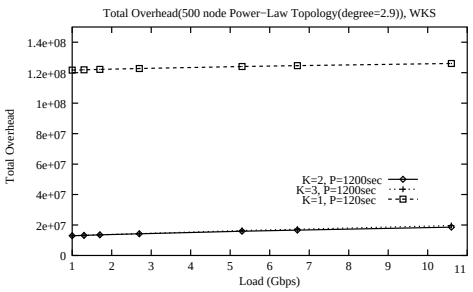

Fig. 10. Total overhead comparison of multiple-path and single path routing

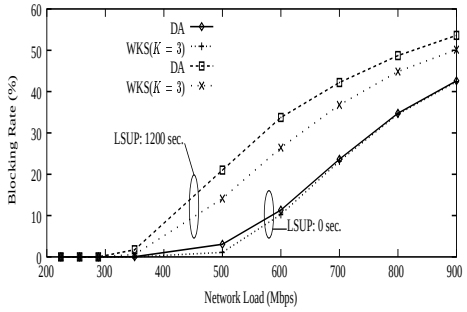

Fig. 11. Dynamic Alternative versus WKS, MCI topology, variable load

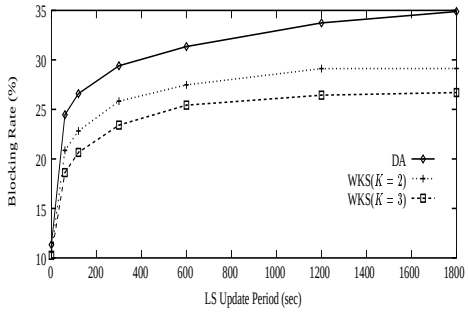

Fig. 12. Dynamic Alternative versus WKS, MCI topology, fixed load

a reservation along the selected path before routing a request. The cost expended during this procedure is called signaling overhead. Fig. 8 shows that WKS with LSUP = 1200 and $K = 2$ or $K = 3$ outperforms the shortest path routing scheme with LSUP = 120. In other words, the multiple-path scheme with inaccurate information wins over the single path scheme with accurate information. This performance benefit is obtained at a cost of increased signaling overhead, as shown in Fig. 9. Nevertheless, as revealed in Fig. 10, the total cost of the multiple-path scheme with inaccurate information is far less than that of the single path scheme. Therefore, we can say that the proposed multiple-path scheme is a scalable solution.

The confrontation of WKS with the *Dynamic–Alternative (DA)* proposed in [20] illustrates why routing schemes that do not account for the inaccuracies in the link state information may perform poorly, even if they appear superior when that information is accurate. This comparison has been done on the MCI topology used in [20]. We observe that if the link state information is accurate (LSUP = 0), DA and WKS exhibit almost the same performance (Fig. 11). However, WKS performs better than DA with outdated link state information (LSUP = 1200 seconds in Fig. 11). For detailed explanation, please refer to [16].

Assume that for a given source–destination pair: the paths in the routing table are denoted $path(i)$ ($1 \leq i \leq K$ for WKS and $1 \leq i \leq 2$ for DA), $bw(i)$ is the bottleneck bandwidth of the ith path, $hop(i)$ is the hop count of the ith path, h_{min} is the hop count of the the minimum hop path, H is the total number of different hop counts for all the K paths. Let n be the hop count of a minimum-hop path. According to [20], a DA path is a Widest–Shortest path with no more than $n + 1$ hops. DA actually is a multiple–path scheme with $K = 2$, $hop(2) - hop(1) = 1$, and $bw(2) > bw(1)$. In the case of LSUP = 0, if there exist paths with h_{min} and $h_{min} + 1$ hops, DA has two alternative paths to choose from. But the situation is more complex for WKS. When the paths are calculated based on accurate link state information, there are in fact only H, not K, paths that can possibly succeed. This is because for all the paths with the same length, if the widest route fails, the others will also fail due to their smaller bandwidth. Only if $H > 1$ can WKS have at least two alternative paths. Thus, if the link state information is accurate, WKS may occasionally provide fewer feasible paths than DA. However, this is not the case when LSUP > 0. For WKS, even if $H = 1$, all K paths can possibly be successful. Besides, for DA, some paths are eliminated by the restriction $bw(2) > bw(1)$, and those eliminated paths could be feasible due to the inaccurate information. Thus, if LSUP > 0, WKS always provides $K \geq 2$ routes to select from while DA provides at most two. As illustrated in Fig. 12, WKS outperforms DA even when $K = 2$.

CONCLUSIONS

We have studied a collection of multiple-path routing schemes and investigated their performance under a diverse set of realistic network topologies. Our study considered a diverse set of traffic conditions and topologies (including realistic topology models) with the intention of uncovering the benefits and limitations of multiple-path routing schemes.

Our experiments have indicated that the proposed approach to routing offers a significant improvement over the single-path approach, especially if the link state information is outdated and inaccurate. The experiments also show that the hop–based algorithms consistently win over the bandwidth–based ones. The best of our routing algorithms, Widest–K–Shortest, also outperforms the *dynamic alternative* when operating with the same value of $K = 2$.

We have demonstrated that a multiple choice of routing paths is one possible remedy for the problem of limited accuracy of the link state information, which is bound to haunt all networks of non-trivial size. Routing solutions that do not take this problem into account will poorly scale to large networks, in which the link state information cannot be updated and propagated too often. As it turns out, it is more important to have a choice at all than to be able to choose from a large selection. Consequently, in terms of computational complexity, our routing algorithms are comparable to those that prefer to stick to a single path. Overall, there is ample evidence that multiple-path schemes can be a scalable solution for QoS routing in environments with inaccurate link state information. Nevertheless, the schemes presented here for the path construction and selection are by no means the only way to consider K alternate paths in QoS routing. Moreover, de-

spite the evidenced advantage of Widest–K–Shortest, other schemes may exist that perform equally well. At the same time, a variety of relevant issues become interesting to study. One example is whether it is possible to approach the performance of selection schemes that select between the shortest paths (as Widest–K–Shortest does), if, instead, we are forced to select between the widest paths due to adminsitrative or contractual restrictions that prohibit a view of the topology (and hence of the hop count) of other network providers.

REFERENCES

[1] G. Apostolopoulos, R. Guerin, and S. Kamat. Implementation and Performance Measurements of QoS Routing Extensions to OSPF. *Proc. INFOCOM'99*, March 1999.

[2] G. Apostolopoulos, R. Guerin, S. Kamat, T. P. A. Orda, and D. Williams. QoS Routing Mechanisms and OSPF Extensions. Internet Request For Comments RFC RFC2676, Internet Engineering Task Force, December 1998.

[3] G. Apostolopoulos, R. Guerin, S. Kamat, A. Orda, and S. K. Tripathi. Intra-Domain QoS Routing in IP Networks: A Feasibility and Cost/Benefit Analysis. *IEEE Network*, September/October 1999.

[4] G. Apostolopoulos, R. Guerin, and S. K. Tripathi. Quality of Service Routing: A Performance Perspective. *Proc. SIGCOMM'98*, ACM, September 1998.

[5] D. O. Awduche et al. RSVP-TE: Extensions to RSVP for LSP Tunnels. Internet Draft draft-ietf-mpls-rsvp-lsp-tunnel-08.txt, February 2001.

[6] A. W. Brander and M. C. Sinclair. A Comparative Study of K-shortest Path Algorithms. *Proc. 11th UK Performance Engineering Workshop*, pages 370–379, Liverpool, September 1995.

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

[8] I. Cidon, R. Rom, and Y. Shavitt. Multi-Path Routing Combined with Resource Reservation. *Proc. IEEE INFOCOM'97*, pages 92–100, April 1997.

[9] B. Davie and Y. Rekhter. *MPLS Technology and Applications*. Morgan Kaufmann, 2000.

[10] D. Eppstein. Finding the K Shortest Paths. *SIAM J. Computing*, 28:652–673, 1999.

[11] R. Guerin and A. Orda. QoS-based Routing in Networks with Inaccurate Information: Theory and Algorithms. *IEEE/ACM Transaction on Networking*, 7(3):350–364, June 1999.

[12] R. Guerin, A. Orda, and D. Williams. QoS Routing Mechanisms and OSPF Extensions. *Proc. GLOBECOM'97*, November 1997.

[13] C. Hsu and J. Y. Hui. Load-Balanced K-Shortest Path Routing for Circuit-Switched Networks. *Proc. IEEE NY/NJ Regional Control Conference*, August 1994.

[14] A. Iwata, N. Fujita, and G. R. Ash. Crankback Routing Extensions for MPLS Signaling. Internet Draft draft-iwata-mpls-crankback-01.txt, July 2001.

[15] J. Jaffe. Algorithms for Finding Paths with Multiple Constraints. *Networks*, 14:95–116, 1984.

[16] Y. Jia, I. Nikolaidis, and P. Gburzynski. Multiple Path Routing in Networks with Inaccurate Link State Information. *Proc. IEEE International Conference on Communications (ICC'2001)*, June 2001.

[17] R. Jain. *The Art of Computer Systems Performance Analysis*. 1991.

[18] B. Jamoussi. Constraint-Based LSP Setup using LDP. Internet Draft draft-ietf-mpls-cr-ldp-05.txt, February 2001.

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

[20] Q. Ma and P. Steenkiste. Quality-of-Service Routing for Traffic with Performance Guarantees. *Proc. IFIP Fifth International Workshop on Quality of Service*, pages 115–126, May 1997.

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

[22] E. Martins, M. Pascoal, and J. Santos. The K Shortest Paths Problem. Technical report, CISUC, June 1998.

[23] N. S. V. Rao and S. G. Batsell. QOS routing via multiple paths using bandwidth reservation. Technical Report 13547, INRIA, October 1998.

[24] A. Shaikh, J. Rexford, and K. G. Shin. Dynamics of Quality-of-Service Routing with Inaccurate Link-State Information. Technical Report CSE-TR-350-97, Dept. of Electrical Engineering and Computer Science, University of Michigan, November, 1997.

[25] Z. Wang and J. Crowcroft. QoS routing for Supporting resource reservation. *IEEE Journal of Selected Areas of Communications*, September 1996.

[26] H. Zhang. Service Disciplines For Guaranteed Performance Service in Packet-Switching Networks. *Proc. the IEEE*, volume 83, October 1995.