Multiple Path QoS Routing

Yanxia Jia, Ioanis Nikoladis, Pawel Gburzynski
{yanxia,yannis,pawel}@cs.ualberta.ca

Abstract

We propose a QoS (Quality of Service) routing scheme in which a connection between a source-destination pair can be assigned to one of several alternative paths. The goal of our approach is to compensate for the inaccuracy of the link state information kept at network nodes. Starting from the well known algorithm by Dijkstra, we develop a collection of K-shortest path routing strategies and investigate their performance under a diverse set of network topologies and traffic conditions. We demonstrate that K-shortest path routing offers a lower blocking probability and more balanced link utilization than traditional schemes based on a single shortest path. Consequently, our approach makes it possible to reduce the frequency of link state exchange, and the incurred bandwidth overhead, without sacrificing the overall performance of the network.

Keywords: Quality of Service, K-shortest path, link state routing, blocking probability

I. INTRODUCTION

A. QoS routing

Traditionally, the problems of routing and providing quality of service (QoS) guarantees were considered separate. It was assumed that the criteria used to determine the "best" path between the source-destination pair could be kept independent of the nature of traffic to be carried along that path. With the introduction of integrated services, networks, including their routing protocols, are being forced to cater to a variety of traffic classes with definite and critical QoS requirements, regarding bandwidth, cost, delay, jitter, loss probability, etc. Consequently, these issues can no longer be ignored by the routing protocols. Their role is not to come up with any "reasonable" path, but with a path that will be able to carry traffic with specific QoS requirements [23].

The basic problem of QoS routing is to find a path satisfying multiple constraints. The metrics involved fall into three types: additive, multiplicative, and concave/bottleneck [25]. Specifically, cost, delay, and delay jitter are of the additive type, loss probability is in the multiplicative category, and bandwidth falls in the concave/bottleneck class. Jaffe [15] demonstrated that the problem of finding optimum paths with delay and cost constraints is NP-complete. Wang *et al.* [25] showed that finding a path subject to constraints on any two or more independent additive or multiplicative metrics in any possible combination is NP-complete as well. Therefore, the problem of multiple-constraint routing appears hard and intractable.

However, as pointed out in [13] and [20], the above metrics are not necessarily independent when some specific scheduling policies are used. In particular, with WFQ-like scheduling, the end-to-end delay and delay jitter depend on the requested bandwidth [13]. Guerin, Orda and Williams [21] showed that with WFQ-like scheduling (and under some reasonable assumptions about the traffic pattern), end to end delay requirements can be translated into bandwidth requirements. Consequently, several studies, e.g., [1, 3, 7, 8], consider bandwidth as the sole metric when investigating the performance of QoS routing. As stated in [22], 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 routing metric. Another metric that we consider is the hop count. The number of hops determines the amount

of resources used along the path, which is an important factor from the viewpoint of efficiency and performance.

To implement QoS routing, network nodes should be aware of the state of links, possibly located several hops away. This calls for a periodic exchange of link state information among the nodes, which involves extra traffic in the network. One problem with QoS routing is therefore the tradeoff 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.

B. Why multiple paths?

Most of the numerous studies of QoS routing, e.g., [1, 9, 10, 11, 12, 19, 25], have been aimed at producing a single optimum path. With this approach, only a single path is established between a source-destination pair, even if there exist some alternative, possibly suboptimal, paths. In case of congestion, this approach is likely to aggravate the problem and may additionally trigger routing oscillations.

With multiple-path routing, the traffic between the same source and destination can be assigned to different paths. 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 of the QoS routing solutions proposed so far are link state based [1, 9, 10, 11, 12, 19, 25]. The main concern of a link state exchange protocol is its poor scalability. 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. This is because, in addition to being collected, information must also be relayed to other nodes. Consequently, frequent exchange of link state information may be prohibitively costly, especially for networks of a considerable 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 link state based QoS routing is how to get acceptable performance in networks with inaccurate link state information.

This problem was investigated to some extent by Guerin and Orda [22]. Given a requested bandwidth, they did not take the bottleneck bandwidth of a path as a reliable metric because it could be false in case of inaccurate state information. Instead, exploiting the inaccurate information, they evaluated the probability of success ("safety") of a path. This way they transformed the routing problem with the bottleneck bandwidth metric (which does not account for inaccurate information), to one with the "safety" metric (which takes inaccurate information into account). 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.

Formally, we address in this paper a bandwidth-constrained multi-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 top 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 what we mean by a "top" path, we have to answer a number of questions before the above specification can be turned into a solution to the routing problem. In particular, we have to say how the K paths are constructed and how they are used to accommodate a traffic stream. "Top" means "shortest" in a certain sense. We develop two categories of K-shortest algorithms, hop-based and bandwidth-based, and correspondingly five path selection algorithms: Best-K-Widest (BKW), Random-K-Widest (RKW), Shortest-K-Widest (SKW), Best-K-Shortest (BKS), and Widest-K-Shortest (WKS). Then we evaluate the performance of those algo-

rithms on a diverse set of network topologies, including well-connected regular and pure random topology, sparsely-connected real topology (the Switched Cluster), and the MCI network [19]. Our results demonstrate that multiple paths provide better routing performance in terms of blocking rate and load balance. We conclude that the hop-based algorithms outperform the bandwidth-based ones, and that the network topology has a paramount impact on the behavior of a routing algorithm.

C. Related work

Recent studies on multi-path routing include [4, 14, 18, 21]. In [21], multiple paths with equal cost are all established and used, but an "equal-cost multi-path" does not necessarily exist for all destinations. Another approach is proposed in [14, 18], where all the multiple paths are used in parallel to route a single traffic stream. In contrast, our scheme is essentially sequential, with resource reservation applied to one path at a time. The parallel multi-path routing scheme [14, 18] 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 [23]. Moreover, those studies do not investigate how the performance of the routing algorithms relates to the accuracy of the link state information.

In [4], the authors present a K-shortest paths algorithm and evaluate its performance. However, that work is based on a fully-connected network and is therefore not applicable to wide area networking, e.g., the Internet. The Bellman-Ford based Widest-Shortest algorithm studied in [12] 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 they also ignore the issue of accuracy of the link state information. In this paper, we compare our WKS variant to DA, with the link state information being accurate as well as inaccurate.

Several papers discuss the algorithms for finding K shortest paths [5, 7]. Our solutions are based on the algorithm presented in [7], which we have modified to find K one-to-all, top, loopless paths.

The remainder of the paper is organized as follows. Section II presents our approach to routing, including the routing mechanism and the implementation model. Detailed algorithms are described in section III, and section IV discusses the simulation environment and some of our results. The conclusions are given in Section V.

II. ROUTING MODEL

A. Link state routing

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 [9]. Obviously, there is a tradeoff between the frequency of state updates (the accuracy of the link state database) and the cost of performing those updates, both in terms of extra network bandwidth and the processing time at the routers. In our model, the link state information is updated periodically, with the *link state update period* (LSUP) varying from 0 to 30 minutes—to create scenarios with different levels of accuracy. We assume 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.

B. Path computation

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 periodically precomputed, with the computation period being equal to LSUP. Thus, when LSUP = 0, the path computation is on-demand. As we mentioned earlier, the QoS requirements of a connection are determined by its bandwidth. Since a routing node cannot predict the actual bandwidth of a connection, we set a bandwidth threshold to prune unsuitable links. This value is carefully chosen, so that the routing algorithm will neither exclude too many feasible paths, nor will it only generate some shortest but infeasible paths (for the hop-based variants).

C. Path selection

When a routing request arrives at a router, the node selects one from the K paths to the destination stored in the routing table. A route check mechanism will determine if this route is feasible, i.e., if there remains enough bottleneck bandwidth along the path. If this is the case, the connection is assumed to have been set up. Otherwise, a second path will be selected according to the same path selection algorithm. If none of the K paths turns out to be feasible, the connection is blocked.

III. ALGORITHMS

A. Constructing K-shortest paths

This is a label setting algorithm based on the Optimality Principle and being a generalization of Dijkstra's algorithm [7]. The 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, the time complexity can be kept at the same level $O(Km)$ [7]. We modify the algorithm to find *one-to-all* loopless paths instead of *one-to-one* non-loopless paths. To reduce the algorithm complexity, we set Max_Hop_Num, the maximum number of hops of a path, to be 16. 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**), or *widest* in terms of the bottleneck bandwidth (**bandwidth-based**). It operates as follows.

Let a DAG (N, A) denote a network with n nodes and m edges, where $N = \{1, ..., n\}$, and $A = \{a_{ij} | i, j \in N\}$. The problem is to find the top K paths from source s to all the other nodes. Define a label set X and an *one-to-many* projection $h : N \to X$, meaning that each node $i \in N$ corresponds to a set of labels $h(i)$, each element of which represents a path from s to i. Each label/path has a major weight and a minor weight. For the hop-based algorithm, the major weight is the inverse of the number of hops and the minor weight is the bottleneck bandwidth of the path represented by this label. Those weight are interchanged for the bandwidth-based algorithm. The minor weight is irrelevant from the viewpoint of the present algorithm (except being computed), but it is used later for path selection (Section III. B).

The algorithm:

- s: the source node
- X: the label set
- b_{vj}: the link bandwidth from v to j
- bw_thrsh: the bandwidth threshold used to trim unqualified links
- $count[v]$: the number of paths from s to v found so far
- $lb0$: the permanent label selected from X, such that $lb0.bw \geq lb.bw$, $\forall\ lb \in X$
- $lb1$: a new label generated from $lb0$
- $lb0.ver$: $h^{-1}(lb0)$, that is, the corresponding node of label $lb0$
- $lb0.bw$: the bottleneck bandwidth of the path from s to $lb0.ver$
- $lb0.parent$: the label which generates $lb0$
- $P_v(count[v])$: the $count[v]$'th path from s to v

$count[i] = 0, \forall\ i \in N$
$lb0 = 1$

$lb0.ver = s$
$lb0.bw = \infty$
$lb0.hops = 0$
$lb0.parent = NULL$
$X = \{lb0\}$
while ($X \neq \emptyset$ and $\exists\ i$ such that $count[i] < K$,
where $0 < i < n$)
do begin
find a permanent label $lb0$ from X, such that
$lb0.bw \geq lb.bw, \forall\ lb \in X$
$X = X - \{lb0\}$
$v = lb0.ver$
$count[v] = count[v] + 1$;
if ($count[v] \leq K$ and $lb0.hops < Max\_Hop\_Num$)
then begin
record the path $P_v(count[v])$ by
following the $lb0 \rightarrow parent$ link
for each $a_{vj} \in A$ /*generate new labels*/
do begin
if (the prospective new label does not result in
a loop and $b_{vj} > bw\_thrsh$
do begin /*generate this label*/
$lb1.ver = j$
$lb1 = lb0 + 1$
$lb1.bw = min(lb0.bw, b_{vj})$
$lb1.hops = lb0.hops + 1$
$lb1.parent = lb0$;
$X = X \cup \{lb1\}$ /*add it to the label set*/
end
end
end
end

Below we demonstrate that the K paths found by the above algorithm for any given destination i are indeed the top K paths from s to i, according to the major weight.

Proof:

Without loss of generality, let us consider the hop-based case. Let the K labels found for node v be lbl_i, $1 \leq i \leq K$. Suppose that by continuing the algorithm, we will get a new path from s to v, whose hop count is smaller than for one of the K found ones. That is, the algorithm will generate a new label lbl_0, such that $\exists\ 1 \leq i \leq K$, $lbl_0.hops < lbl_i.hops$. Naturally we get $lbl_0.parent.hops < lbl_i.parent.hops$ (because $l.parent.hops = l.hops + 1, \forall\ l \in X$). This implies that $lbl_0.parent$ must have been marked as a permanent label earlier than $lbl_i.parent$, because in each step of the algorithm, only the best label is marked as a permanent label. Consequently, lbl_0 should have been generated earlier that lbl_i, which is a contradiction. contradiction. A very similar reasoning will take care of the bandwidth-based algorithm.

B. Path selection algorithms

Below we list five path selection algorithms resulting from two different major criteria and different ways of applying the minor criteria. Although the names Shortest-K-Widest and Widest-K-Shortest resemble Shortest-Widest from [25] and Widest-Shortest from [12], our algorithms are quite different from those previously proposed solutions.

- Best-K-Widest (BKW): From the K widest paths, select the one whose bandwidth best fits the connection request.
- Random-K-Widest (RKW): From the K widest paths, select one at random.
- Shortest-K-Widest (SKW): From the K widest paths, select the one with the least number of hops. The difference between this algorithm and the Shortest-Widest (SW) algorithm from [25] is that SW finds only one optimum path, while SKW finds top K paths. SW uses the "shortest-widest" constraint during the path construction phase. That is, when multiple labels with the same bandwidth are found, only the shortest one is permanently labelled and the others are deleted. This is why SW can only find one optimum path. In contrast, SKW keeps 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.
- Best-K-Shortest (BKS): From the K shortest paths, select the one whose bandwidth best fits the connection request.
- Widest-K-Shortest (WKS): From the K shortest paths, select the one with the largest bandwidth. Although the Widest-Shortest (WS) algorithm from [12] also generates multiple paths for a given destination, it is different from WKS in that it provides fewer choices for selecting short paths, and is thus likely to require more resources. In particular, it uses

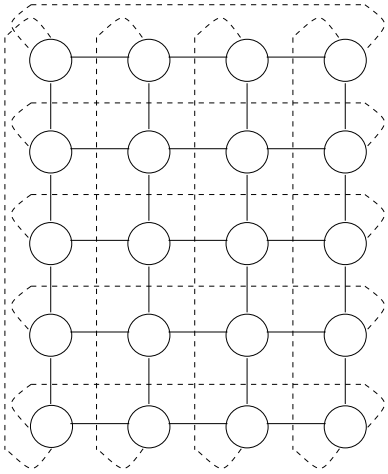


Figure 1: The $5X4$ Torus topology

an upper bound for the hop count, and for each hop count, it can only keep one widest path. Consequently, it ignores the equal-hop-count multiple paths. By selecting from the top K shortest paths, WKS has more choices. Besides, WS requires that the $i+1$th path be "wider" than the ith 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 impairs its performance when the link state information is inaccurate.

IV. SIMULATION EXPERIMENTS

A. Topologies

To investigate the behavior of our routing scheme, we consider four different network topologies, each consisting of around 20 nodes. Besides two artificial configurations, i.e., the Torus (see Figure 1), and the purely random topology [8], we also use the MCI network [19] and the Switched Cluster topology [19] as examples of real-life networks. The Torus structure is regular with the same degree (4) of every node. For both artificial networks, the capacity of all links is assumed to be the same: 45Mbps for the Torus and 100Mbps for the random network.

B. Traffic patterns

We simulate the network behavior for the period of 15 hours under a connection-oriented load. All connections must receive the bandwidth they ask for to be successfully set up. The overall load of the network is determined by the connection arrival process, the holding time distribution, and the requested bandwidth. Specifically, the connection arrival process is Poisson. Following [24], the connection holding time is sampled from a lognormal distribution (considered a more accurate replacement for the traditionally accepted exponential distribution) with the mean of 3 minutes. The requested bandwidth is determined by the *composition* of the connection, which can be of two types. Type A consists of 60% audio traffic, whose bandwidth is uniformly distributed between 16Kbps and 64Kbps, and 40% video traffic, whose bandwidth is between 1Mbps and 5Mbps. Type B consists of pure video traffic. Additionally, the arriving traffic can be evenly distributed over the entire network, or biased, i.e., favoring some source-destination pairs. For a better comparison, the hot source-destination pairs on the MCI network and the Cluster are the same as in [19]. The overall intensity of the traffic is tuned by adjusting the average arrival rate.

C. Performance metrics

We use two performance measures. The first of them, the *bandwidth blocking rate (BW_BR)* [19, 20], is defined as follows:

$$BW\_BR = \frac{\sum blocked\_bandwidth}{\sum requested\_bandwidth} \qquad (1)$$

A connection can be blocked somewhere in the network for any of two reasons. One possibility is that there is no path available with enough bandwidth to satisfy the request. The other reason is the incorrect path selection due to the inaccurate (outdated) link state information. The latter situation can in turn occur in two basic flavors. It may happen that no path is found to have sufficient bandwidth, while in fact there exist

such paths. Or a path may be found that appears to have enough bandwidth, but it later turns out to be insufficient. Clearly, incorrect path selection can increase the blocking rate. The second of our performance measures attempts to capture how the blocking rate is influenced by multiple paths.

We define the Coefficient of Variation of the link utilization as:

$$C_v(LU) = \frac{Std(LU)}{u(LU)}, \quad (2)$$

where LU stands for link utilization, $Std(LU)$ is the Standard Deviation of the link utilization, and $u(LU)$ is the mean of the link utilization. $C_v(LU)$ reflects the relative dispersion of the load in the links regardless of the link utilization average.

V. RESULTS

The goal of our simulation experiments is to investigate the performance of the path selection algorithms listed in Section III.B under different network topologies and traffic conditions. We are primarily interested in a comparison of K-shortest path routing with single-path routing in the face of inaccurate link state information.

A. The impact of K

A.1 Blocking rate

Figures 2, 3, 4 and 5 show how the value of K influences the blocking rate in different networks. At least for some algorithms (SKW appears to be the winner), and if LSUP is not too small, any K bigger than 1 drastically reduces the blocking rate. Notably, the difference between $K = 2$ and $K = 3$ is much smaller than the difference between $K = 1$ and $K = 2$, which indicates that it is important to have a choice, but that choice need not be very big. Indeed, our experiments indicate that increasing K beyond 3 brings about a rather moderate improvement compared to the $K = 3$ case. Consequently, to reduce the number of curves, we will use $K = 3$ as a standard case of multiple paths.

The difference between $K = 1$ and $K > 1$ tends to increase as LSUP becomes bigger, which means that multiple-path routing is better suited to cope with increasing inaccuracies in the link state information than single-path routing. This also means that multiple-path routing better scales to the increasing network size.

The observation that small $K > 1$ suffices to produce a significant improvement over the $K = 1$ case 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.

A.2 Load balance

Our primary intuition behind multiple paths was that with several alternative routes, we would be able to spread the load more evenly over the network. Thus, in this section, we investigate the impact of multiple paths on load balance.

In a series of experiments run on a $5X4$ Torus, we calculate $C_v(LU)$ for all routing algorithms and three values of K. As explained earlier, smaller $C_v(LU)$ means low variations in link utilization. With the capacity of all links being identical, a lower value of $C_v(LU)$ indicates more balanced link utilization. Figure 6 shows that for each algorithm, bigger K implies lower $C_v(LU)$. This naturally correlates with the lower blocking probability. In fact, the trend of the blocking probability closely resembles that of $C_v(LU)$. Also in this case, hop-based algorithms turn out to be less sensitive to inaccuracies in the link state information.

B. A comparison of the algorithms

B.1 The blocking rate

Figure 7 plots the blocking rate as a function of LSUP for the five algorithms in the four topologies considered in our study. There is a clear gap between the blocking rates of the bandwidth-based and hop-based algorithms. Under the same condi-

tions, i.e., the same topology, the same traffic load, and the same LSUP, the latter exhibit a significantly lower blocking rate. The reason behind this is that the hop-based algorithms tend to occupy fewer resources (links) for each request, thus being able to accommodate more requests than the bandwidth-based algorithms.

We can also see that the hop-based algorithms find it easier to put up with inaccuracies in the link state information. The difference in performance of the two classes of algorithms becomes more pronounced with increasing LSUP, especially for the Torus, the MCI network, and the Cluster.

B.2 The impact of topology

Figure 7 also reveals how differences in the network topology affect the performance of our routing algorithms. To make the comparison fair, we adjust the topology parameters so that for the same traffic conditions, the same inaccuracy level of link state information, and the same algorithm, i.e., Widest-K-Shortest, the observed blocking rates are nearly the same (10%) for all four topologies. This way, we make the "blocking potential" of the four networks the same. Simulation experiments carried out under these circumstances demonstrate that when the blocking curve reaches a stable point, e.g., LSUP = 1800, the discrepancy in the blocking rate between the two classes of algorithms reaches as high as 36% in the Torus and only 8% in the MCI network. There is also a big difference between the two real-life topologies, i.e., 22% in the Cluster and 8% in the MCI network. This demonstrates that although these topologies have the same "blocking potential," their structural differences impact the behavior of our routing algorithms to a different degree.

C. Comparison with related work

In [19], Ma introduced and compared several routing algorithms, advocating the *dynamic alternative* (DA) as the best-performing solution under QoS constraints. We make a comparison between the DA and our WKS algorithm under identical traffic conditions (100% biased video traffic) and on the same network topology (the MCI network). The biased traffic conditions can be viewed as creating extremities in the network, and thus providing an interesting and challenging environment for comparing protocols that purport to cope with unfriendly scenarios.

We observe that if the link state information is accurate, DA and WKS exhibit almost the same performance (Figure 8(a)). However, WKS performs better than DA with outdated link state information (Figures 8(b) and 8(c)). Before explaining why this is the case, let us introduce some notation. Assume that for a given source-destination pair,

- the paths in the routing table are denoted $path(i)$, where $1 \leq i \leq K$ for WKS and $1 \leq i \leq 2$ for DA,
- $bw(i)$ is the bottleneck bandwidth of the of 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.

As described in [19], "Let n be the hop count of a minimum-hop path when the network is idle. A dynamic-alternative path is a widest-shortest path with no more than $n + 1$ hops." DA actually is a multi-path scheme with $K = 2$, $hop(2) - hop(1) = 1$, and $bw(2) > bw(1)$. In 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. If $H = 1$, WKS has only one path that can be successful, even though it constructs K paths. 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. This is because the link state information is inaccurate, and a "narrower" path in the routing table could turn out to be wider than the "widest" path. 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 link state information. Thus, if LSUP > 0, WKS always provides $K \geq 2$ routes to select from while DA provides at most two.

VI. Conclusions

We have presented a K-shortest path QoS routing scheme for connection-oriented networks and the resulting two classes of routing algorithms: bandwidth-based and hop-based. Our experiments indicate that the K-shortest path routing approach offers much better performance than other schemes, especially if the information regarding the state of links in the network is obsolete and inaccurate. The reason for this is that multiple paths tend to disperse the traffic load over the network, thus reducing congestion.

Within our proposed scheme, the class of hop-based algorithms wins over the bandwidth-based algorithms in terms of the blocking rate, load balance, and sensitivity to the inaccuracy of the link state information. The hop-based algorithms can be recommended for networks in which link state information cannot be updated too frequently, e.g., because of the network size. Those algorithms scale much better to the increasing network size than other QoS routing solutions, including the dynamic alternative (DA) introduced in [19].

REFERENCES

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

[2] A. Shaikh, J. Rexford, and K. G. Shin, ""Efficient precomputation of quality-of-service routes," In *Proceedings of Workshop on Network and Operating Systems Support for Digital Audio and Video*, July 1998, pp. 15-27.

[3] A.W. Brander, M.C. Sinclair, "A Comparative Study of K-shortest Path Algorithms", In *Proceedings of 11th UK Performance Engineering Workshop*, Liverpool, pp.370-379, September 1995,.

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

[5] D. Eppstein, "Finding the K Shortest Paths", *SLAM J. Computing* 28:0(1999) pp. 652-673.

[6] E.Crawley, R. Nair, B. Rajagopalan, H. Sandick, "A Framework for QoS-based Routing in the Internet", RFC2386, August 1998.

[7] E.Q.V. Martins, M.M.B. Pascoal and J.L.E. Santos, "The K Shortest Paths Problem", Research Report, CISUC, June 1998.

[8] E.W. Zegura, K. Calvert and S. Bhattacharjee, "How to Model an Internetwork", In *Proceedings of IEEE Infocom '96*, San Francisco, CA.

[9] G. Apostolopoulos, R. Guerin, and S. K. Tripathi, "Quality of Service Routing: A Performance Perspective", *SIGCOMM'98*, Vancouver, BC, September, 1998.

[10] G. Apostolopoulos, R. Guerin, and S. Kamat, "Implementation and Performance Measurements of QoS Routing Extensions to OSPF", *IEEE INFOCOM'99*, New York, NY, March 1999.

[11] 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.

[12] G. Apostolopoulos, R. Guerin, S. Kamat, A. Orda, T. Przygienda, and D. Williams, "QoS Routing Mechanisms and OSPF Extensions", RFC, December, 1998.

[13] H. Zhang, "Service Disciplines For Guaranteed Performance Service in Packet-Switching Networks", In *Proceedings of the IEEE*, 83(10), Oct 1995.

[14] I. Cidon, R. Rom, and Y. Shavitt. "Multi-Path Routing Combined with Resource Reservation". *IEEE INFOCOM'97*, April 1997, Kobe, Japan, pp. 92–100.

[15] J. Jaffe, "Algorithms for Finding Paths with Multiple Constraints", *Networks*, Vol 14, pp95-116.

[16] J. Moy, "OSPF Version 2", RFC 2328, Apr. 1998

[17] K. Calvert, M. Doar and E.W. Zegura, " Modeling Internet Topology", *IEEE Communications Magazine*, June 1997.

[18] N. S. V. Rao, S. G. Batsell, QOS routing via multiple paths using bandwidth reservation, Technical Report No. 13547, October 1998, an earlier version appeared in INFOCOM'98.

[19] Q. Ma, *"Quality-of Service Routing in Integrated Services Networks"*, Ph.D. thesis, CMU-CS-98-138, January, 1998.

[20] Q. Ma and P. Steenkiste, "Quality-of-Service Routing for Traffic with Performance Guarantees", In *Proceedings of IFIP Fifth International Workshop on Quality of Service*, pp. 115-126, Columbia University, New York, May 1997

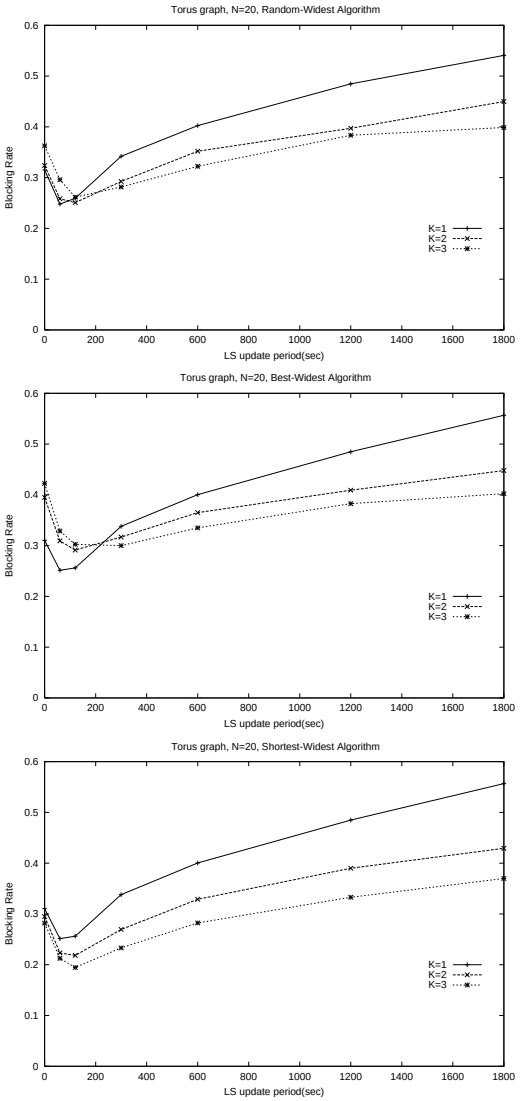
[21] R. A. Guerin, A. Orda, and D. Williams, "QoS Routing Mechanisms and OSPF Extensions", In *Proceedings of 2nd Global Internet Miniconference (joint with Globecom'97)*, Phoenix, AZ, November 1997.

[22] R. Guerin and A. Orda, "QoS-based Routing in Networks with Inaccurate Information: Theory and Algorithms." *IEEE/ACM Transaction on Networking*, Vol. 7, No. 3, June 1999, pp. 350-364.

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

[24] V. A. Bolotin, Modeling Calling Holding Time Distributions for CCS Network Design and Performance Analysis, *IEEE JSAC*, 12(3):433-438, April 1994.

[25] Z. Wang and J. Crowcroft, "QoS routing for supporting resource reservation", *IEEE Journal of Selected Areas of Communications*, September 1996, http://www.bell-labs.com/user/zhwang/pub.html.



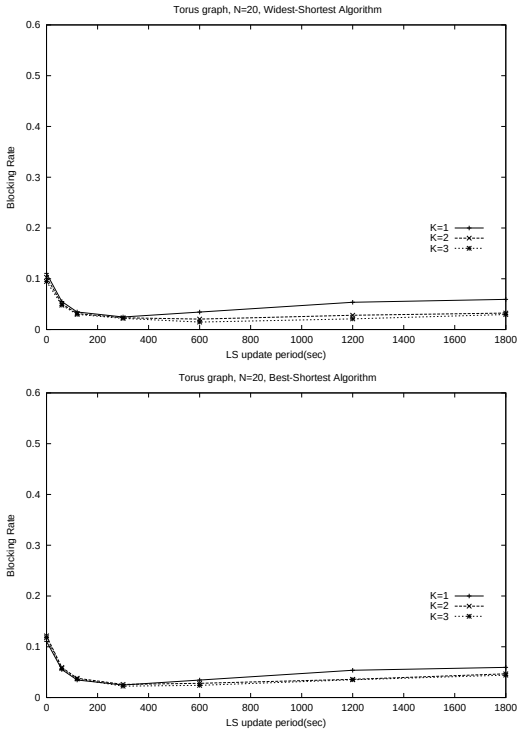
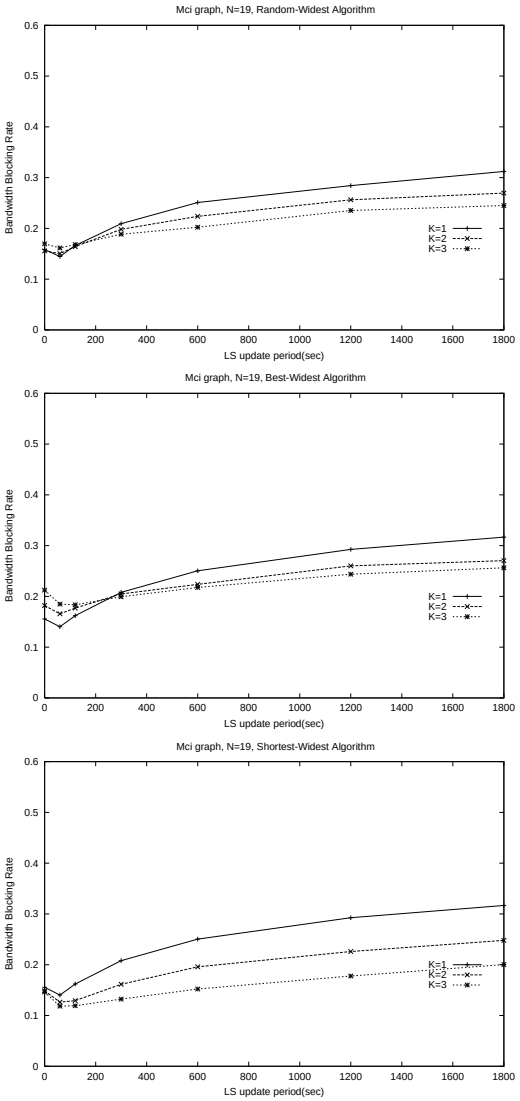


Figure 2: Torus, 100% evenly distributed video traffic, Network load = 280Mbps



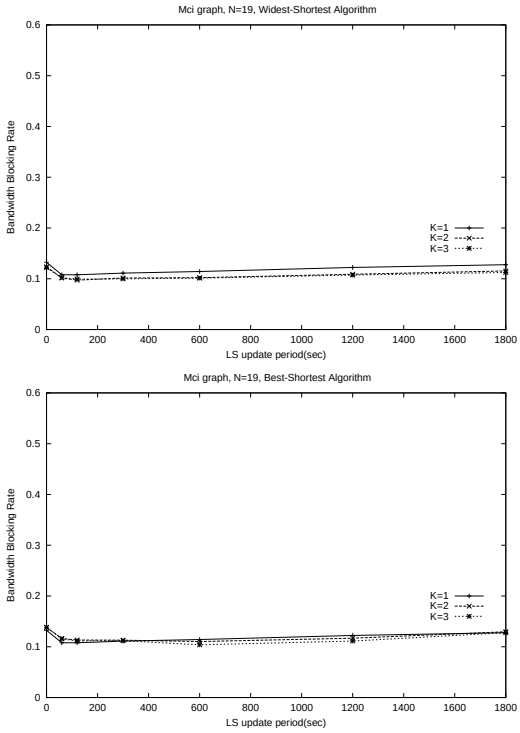
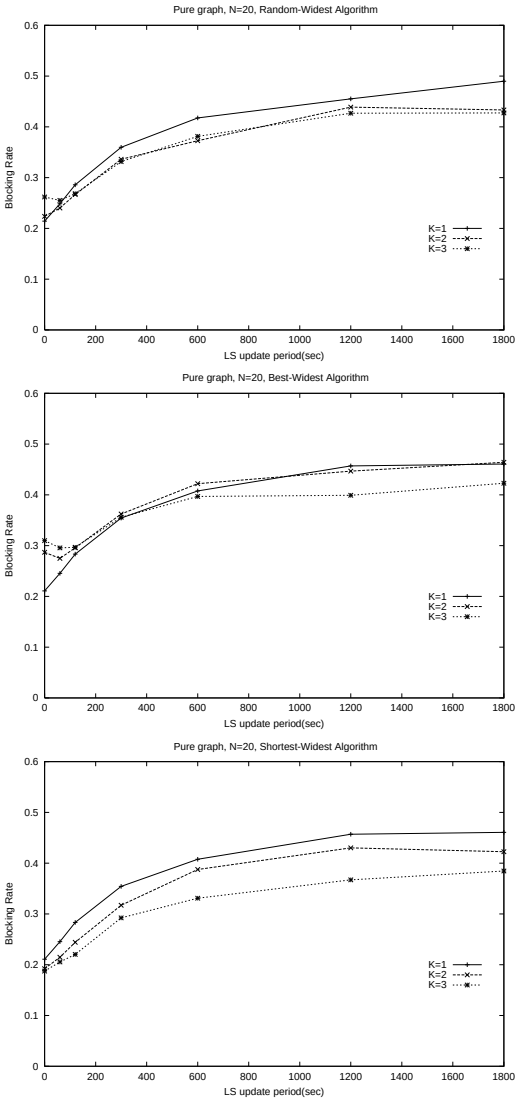


Figure 3: MCI, 100% evenly distributed video traffic, Network load = 620Mbps



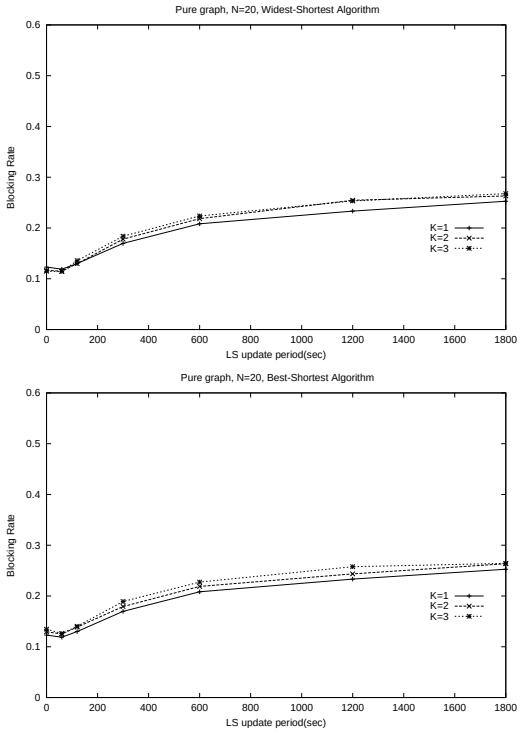
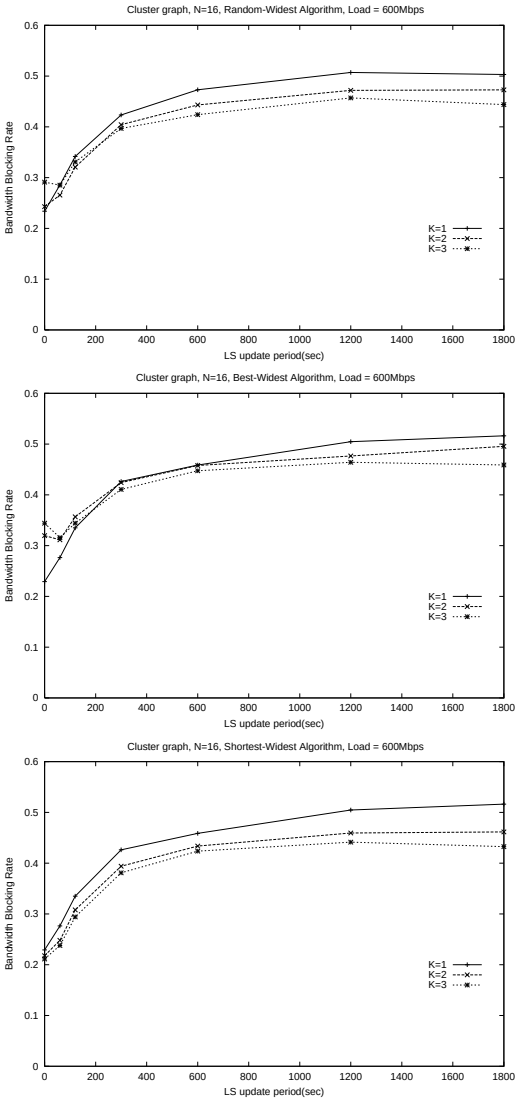


Figure 4: Pure random, 100% evenly distributed video traffic, Network load = 530Mbps



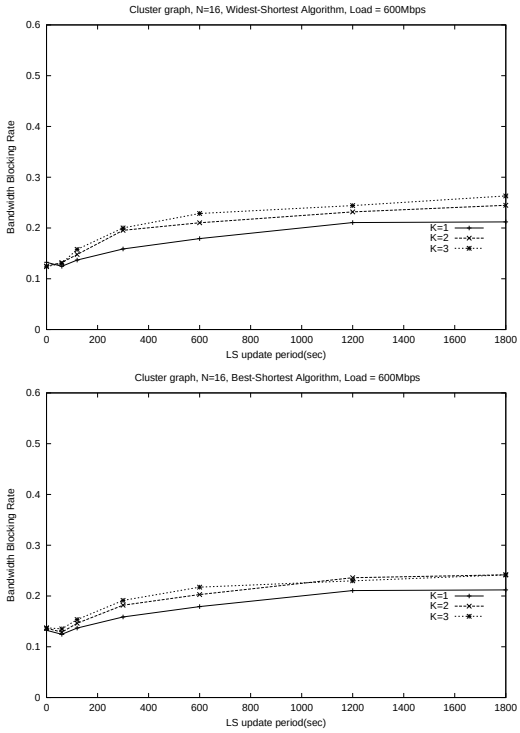
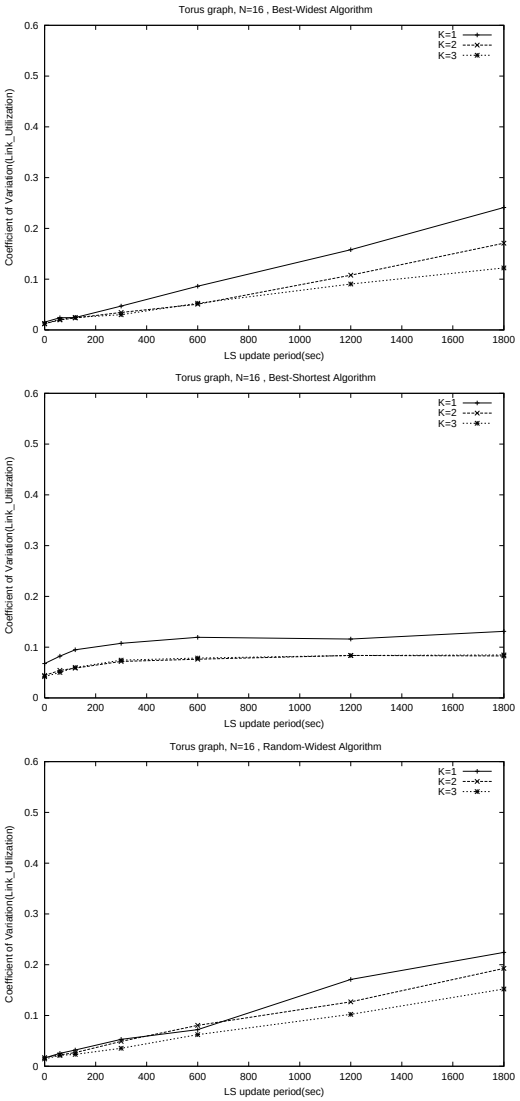


Figure 5: Cluster, 100% evenly distributed video traffic, Network load = 600Mbps



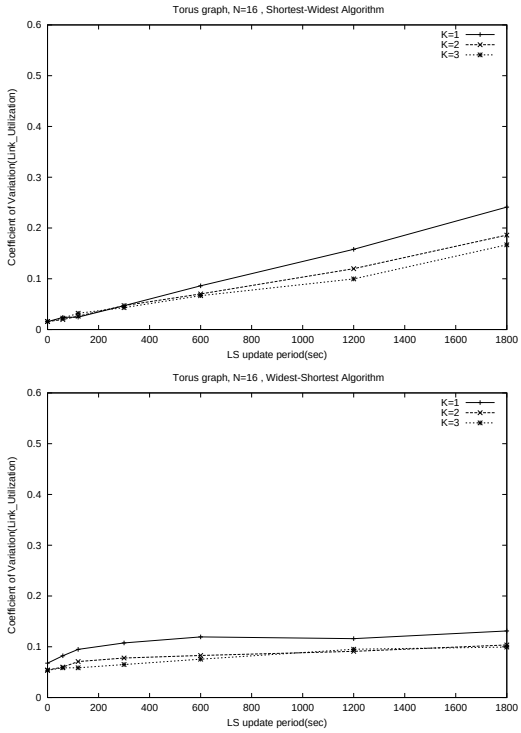


Figure 6: Torus, 100% evenly distributed video traffic, Network load = 280Mbps

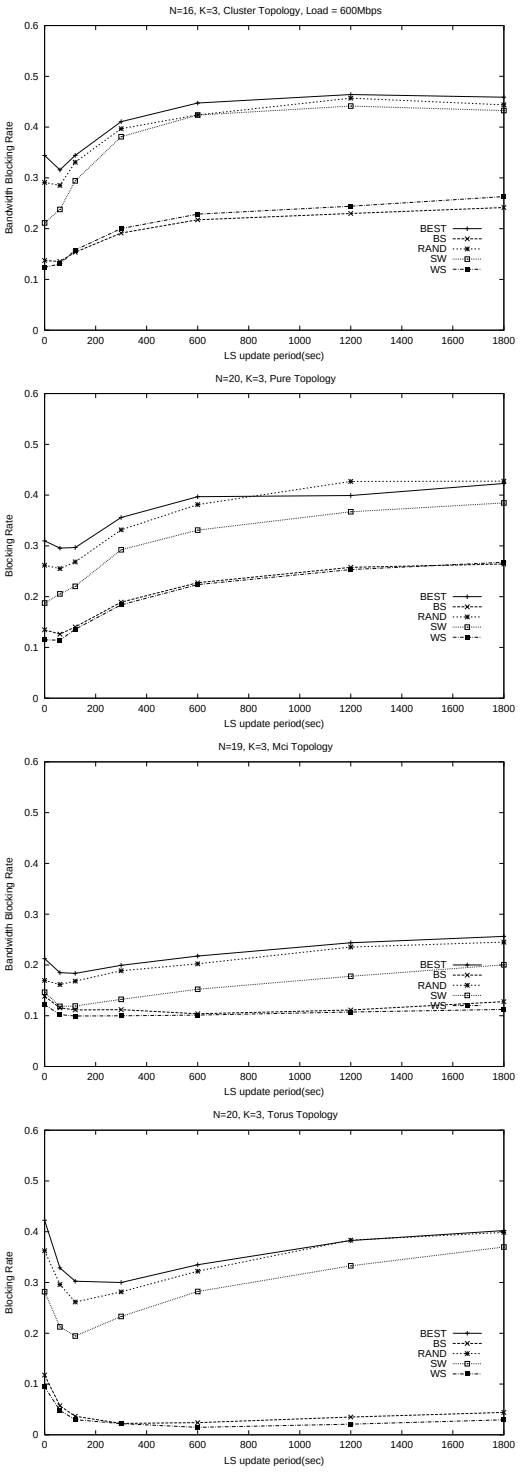


Figure 7: Comparison among 5 algorithms in Cluster, pure random, MCI and Torus

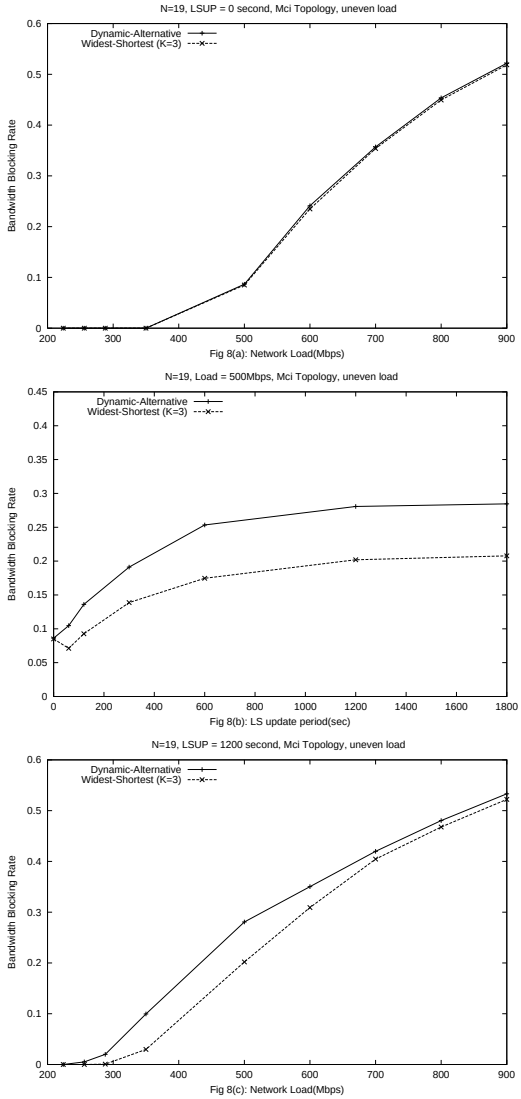


Figure 8: Comparison between the DA and WKS algorithms