Routing Traffic with Quality-of-Service Guarantees in Integrated Services Networks

Qingming Ma
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134
qma@cisco.com

Peter Steenkiste
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
prs@cs.cmu.edu

Abstract

Transmission of interactive multimedia streams requires stringent Quality-of-Service (QoS) guarantees in delay, delay jitter, and bandwidth, which imposes strict resource constraints on paths being used. While the general problem of finding a path that meets multiple QoS constraints is NP complete, we have shown that such a path can be found in polynomial time if the network service disciplines are rate proportional. However, two important issues remain unsolved. One issue is how to select among all feasible paths an efficient one that achieves high network throughput. We identify four optimality criteria and for each of them we propose a polynomial algorithm that selects paths that meet the criterion. Our simulation results show that, while the widest-shortest path performs best for heavy loads, the shortest-delay path has a performance edge for light loads. Surprisingly, the path requiring to reserve the minimum bandwidth performs the worst for both heavy and light loads. We also evaluate the influence of these routing algorithms on the throughput of best effort traffic. Our results show that, using the shortest-widest path algorithm for routing guaranteed traffic can result in relatively low throughput for best-effort traffic, while the other three algorithms result in comparable performance. The other important issue is that, although these algorithms are polynomial, they have much higher computational complexity than the Bellman-Ford algorithms used in today's Internet. We propose approximation algorithms that have the running time of a constant factor of the Bellman-Ford algorithm without sacrificing network throughput. Overall, the approximation algorithms that select the shortest-delay path results in consistently high network throughput.

1 Introduction

The advent of high speed networking has introduced opportunities for real-time multimedia applications such as video conferencing and Internet telephony. These applications have stringent Quality-of-Service requirements (QoS) in terms of bandwidth, delay, delay jitter, and loss rate. To provide guaranteed QoS, new network service models have been defined in both the IP and ATM community [29, 1] and new resource management mechanisms that support these service models are being developed [6, 4, 36, 23, 34, 32, 10]. QoS routing is one such mechanism with two goals: selecting feasible paths that meet QoS constraints and making efficient utilization of the network resources. To achieve the goals, both routing protocols and the routing algorithms must be developed. While QoS routing protocols have received much attention in both the ATM forum and the IETF QoS Routing group, QoS routing algorithms have not been well understood.

Selecting a feasible path that meets multiple QoS constraints (e.g., delay and delay-jitter) is in general computationally intractable [7, 33]. However, this result is obtained under the assumption that QoS constraints (e.g., delay, delay jitter, and bandwidth) are not related and that queueing delay is known *a priori* regardless of the burstiness of the traffic source and the amount of bandwidth to reserve. In our earlier study [21], we observed that that QoS constraints are interrelated in a way that is determined by the network scheduling discipline and that queueing delay is determined by the amount of bandwidth to reserve and the burstiness of the traffic source. By exploiting these dependencies in a network whose service disciplines are rate proportional (e.g., Weighted Fair Queueing), we developed polynomial algorithms for selecting feasible paths with delay, delay jitter, and bandwidth constraints. The resulting routing algorithms iterate the Bellman-Ford (IBF) algorithm over different bandwidth values of all links in the network.

In this paper, we move further by addressing two important issues that have not been well understood in QoS routing. First, the IBF algorithm presented in [21] finds a feasible path as long as it exists, but more than one feasible path is often available. This raises the question of which feasible path to select to achieve high network throughput. We identify four optimality criteria that can be used when selecting an efficient path. These criteria strike a different balance between two competing goals: minimizing the resource consumption of individual flows and spreading the network load. Based on the IBF algorithm, we propose routing algorithms that select paths with these different optimality criteria. We evaluate these algorithms through an extensive simulation study, using realistic topology, traffic model, and periodic routing information distributions. Our evaluation uses the call blocking rate as the primary performance metric. However, in a network that supports both guaranteed traffic and best-effort traffic, the choice of a routing algorithm for higher priority guaranteed traffic can influence the throughput of lower priority best-effort traffic, since guaranteed sessions can take away resources used by on-going best-effort traffic. The throughput for best-effort traffic becomes a more important performance metric when the volume of guaranteed traffic is low and the call blocking rate is not an issue. Our results show that while the widest-shortest path algorithm performs well when the load is heavy because it conserves resources, the shortest-delay path algorithm has a performance edge when the load is light because it distributes the network load more evenly than other algorithms. Surprisingly, the algorithms focusing on minimizing bandwidth reservation perform the worst.

The second issue we address in this paper is that the computational complexity of the IBF based algorithms are much higher than the Bellman-Ford algorithm, although they are polynomial. We develop an approximation scheme to these algorithms that result in algorithms with the computational complexity a constant factor of the Bellman-Ford algorithm. Our simulation results show that these approximation algorithms run almost ten times faster than the original algorithms for a MCI Internet backbone topology. Most importantly, these algorithms do not sacrifice the network throughput. Overall, the approximation to the shortest delay path algorithm performs consistently well for different topologies and traffic load.

Our results demonstrate that QoS routing is desirable because of significant throughput improvement by using carefully selected routes and QoS routing is feasible because QoS routes can be selected as fast as the Bellman-Ford algorithm up to a constant factor.

In the remainder of this paper, we briefly review in Section 2 the IBF algorithm discussed in [21] that selects feasible paths for guaranteed traffic in polynomial time. In Section 3 we identify optimization criteria and present corresponding routing algorithms. Sections 5 and Section 6 examine the performance of these routing algorithms, and Section 7 present and evaluate our approximate algorithms. Related work is discussed in Section 8 and we summarize in Section 9.

2 Selecting Feasible Paths

As discussed in [21], by exploiting the relationship between QoS constraints introduced by the rate-proportional service disciplines, we are able to develop polynomial QoS routing algorithms that select paths subjected to delay, delay jitter, and/or buffer space constraints. In this section, we summarize the main results obtained in [21].

2.1 Rate-proportional Service Disciplines

A broad class of service disciplines aimed at providing per-flow performance guarantees have been proposed. Examples include Virtual Clock (VC) [35], Weighted Fair Queueing (WFQ) [6, 23], Worst-case Weighted Fair Queueing ($\mathrm{WF}^2\mathrm{Q}$) [2], Self Clocked Fair Queueing (SCFQ) [9], and Frame-Based Fair Queueing [31]. These service disciplines ensure that flows sharing an output link get their proportional shares of the link capacity. As a result, the end-to-end queueing delay, the delay jitter, and required buffer space in each network node is determined by the bandwidth reserved for the flow and the characteristics of the traffic source.

Given a path $\mathbf{p}$ of n hops with link capacity C_i at hop i, and a traffic source constrained by the leaky bucket $\langle\sigma, b\rangle$, where σ is the average token rate and b is bucket size, provable bound on end-to-end delay, delay jitter, and buffer space requirements at the h-th hop are given by [34, 32]:

$$D(\mathbf{p}, r, b) = \frac{b}{r} + \frac{n \cdot L_{\max}}{r} + \sum_{i=1}^{n} \frac{L_{\max}}{C_i} + \sum_{i=1}^{n} \mathrm{prop}_i \quad (1)$$

$$J(\mathbf{p}, r, b) = \frac{b}{r} + \frac{n \cdot L_{\max}}{r}. \quad (2)$$

$$B(\mathbf{p}, b, j) = b + h \cdot L_{\max}. \quad (3)$$

where r ($r \geq \sigma$) is the amount of bandwidth to be reserved, $L_{\max}$ is the maximal packet size in the network, and prop_i is the propagation delay. These equations relate delay, delay jitter, and buffer space to the bandwidth and the leaky bucket of the traffic source.

2.2 Iterative Bellman-Ford Algorithm

The main results of [21] are summarized in the following theorem. We refer the reader to [21] for detailed proofs and algorithms.

Theorem 1 *The QoS routing problem of finding a path with delay, delay-jitter, and/or buffer space constraints is solvable in polynomial time. If the bandwidth to be reserved is known* a priori*, a slightly modified version of the Bellman-Ford algorithm can solve it in $S = O(m \cdot L)$, where m is the number of nodes and L the number of links in the network. If the bandwidth to be reserved is unknown, an algorithm that iterates the modified version of the Bellman-Ford algorithm can solve it in $E \cdot S$, where E is the number of all possible residual bandwidths of links in the network.*

We briefly present the intuition behind the IBF algorithm. The simplest case is when the amount of bandwidth r to reserve is known *a priori*. In that case, a feasible path can be found using any shortest path algorithm by using the following link cost function:

$$d(j) = \frac{L_{\max}}{r} + \frac{L_{\max}}{C_j} + \mathrm{prop}_j \quad (4)$$

```
for k ← 1 to e do                 /* over different values of link residual bandwidth */
    for h ← 1 to m − 1 do         /* over hop count, m is the number of nodes */
        for (x, y) ∈ L do         /* over all links (x, y) ∈ G */
            Relax(x, y, v_k, ...)
    ......                        /* the rest of the program */
return
```

Figure 1: A sketch of the pseudo-code for the IBF algorithm

In practice, the bandwidth r to be reserved is not known *a priori* and should be determined by the routing algorithm. This creates a dependency between r, the delay, and the path being selected, and a simple shortest path algorithm no longer works. However, it is clear that the path with the shortest delay will be feasible, if a feasible path exists. Such a path requires reserving the maximum reservable bandwidth (mrb) on the path

$$\mathbf{mrbp} = \min\{R_j \mid j \in \mathbf{p}\},$$

to achieve the minimum delay, where **p** is the path and R_j the residual bandwidth, i.e., the amount of reservable bandwidth of link j. Note that the $\mathbf{mrbp}$ must be equal to R_j for some $j \in \mathbf{p}$,

This observation leads to an algorithm for finding a shortest-delay path, and thus a feasible path if one exists. We simply iterate over all different residual link bandwidths R in the network, and in each iteration we assume that the bandwidth R needs to be reserved and find the minimum-delay path using the cost function of Equation 4. If none of these paths is feasible, no feasible path exists. Otherwise we have a solution, and we can use Equation 1 to determine how much bandwidth actually has to be reserved. When the Bellman-Ford shortest path algorithm is used in each iteration, we call this algorithm the Iterative Bellman-Ford algorithm, or IBF algorithm. A sketch of the pseudocod for the IBF algorithm is shown in Figure 1; the procedure Relax$(x, y, v_k, \ldots)$ updates the paths if the distance of the current shortest path from the source to x plus the distance between x and y is smaller than the distance of the current shortest distance path from the source to y. If not only delay, but also jitter and buffer size constraints have to be met, a similar algorithm can be used. The only difference is that a hop count constraints have to be added to ensure jitter and/or buffer size constraints.

We do not consider in this paper the more general case in which a different amount of bandwidth can be reserved on each link in this paper. In that case, QoS routing is NP-complete [19].

3 Selecting Efficient Paths

In practice, more than one feasible path is often available. This raises the question of which path to use to achieve higher network throughput (or lower blocking rate). We propose four different optimality criteria that can be used in path selection and propose the routing algorithms that select paths with different optimality properties.

3.1 Optimality Criteria

What path selection criterion will minimize the call blocking rate for guaranteed traffic is an open question. A first goal could be to minimize the resource utilization of selected paths. For guaranteed traffic, this can be achieved by selecting the feasible path with the **minimum hop count** or that requires reserving the **minimal bandwidth**. An alternative goal is to try to distribute the load evenly through the network. Since mrb can be viewed as an indication of the load conditions (for guaranteed traffic), selecting the **minimum load path**, i.e. the path with maximum mrb, would emphasize distributing load in the network. Finally, we can select a path with **minimum end-to-end delay**, if the mrb is reserved (this does not imply that the mrb has to be reserved). This last criterion is motivated by the form of Equation 1: it suggests that a path low end-to-end delay will in general have few hops and a high mrb, so this criterion strikes a balance between minimizing resource use and distributing load.

Different combinations of these optimality criteria lead to the following path selection criteria:

- **Minimum-bandwidth path**—a feasible path that requires the reservation of the minimum amount of bandwidth. If there is more than one choice, the one with the minimum hop count is selected.
- **Widest-shortest path**—a feasible path with the minimum hop count. If there is more than one path with the minimum hop count, the one with the maximum reservable bandwidth is selected.
- **Shortest-widest path**—a feasible path with the maximum reservable bandwidth. If there are several such paths, the one with the minimum hop count is selected.
- **Shortest-delay path**—a feasible path giving the minimal end-to-end delay if the maximal reservable bandwidth is reserved. If there are several such paths, the one with the minimum hop count is selected.

Clearly other path selection criteria are possible. We selected these algorithms because they represent a broad spectrum of different tradeoffs between resource conservation and network load distribution as is shown in Figure 2.

minimum-bandwidth	widest-shortest	shortest-delay	shortest-widest
⟵ *resource conserving*			*load distribution* ⟶

Figure 2: Tradeoffs for path selection criteria

3.2 Algorithms

We present algorithms that can identify paths that meet the above optimality criteria. Recall that the IBF algorithm has two levels of iterations: the outer loop (line (a) in Figure 1) over different values of link residual bandwidth and the inner loop (line (b) in Figure 1) over the hop count. For every value of link residual bandwidth and every hop count, a feasible path may be found. We store all these paths obtained in an array as is shown below. We will now show that paths corresponding to the different optimal criteria can be found from the feasible paths recorded in the table.

bandwidth value	hop 1	$\cdots$	hop $m-1$
v_1	$\mathbf{p}_{1,v_1}$		$\mathbf{p}_{m-1,v_1}$
...			
v_e	$\mathbf{p}_{1,v_e}$		$\mathbf{p}_{m-1,v_e}$

The *shortest-delay* path can be simply be read from the table after all iterations have been completed. To find a *minimum-bandwidth path*, we compare all feasible paths in the table, and select the one requiring the reservation of the minimum bandwidth (use Equation 1). We claim that there is no other path requiring less bandwidth that meets the delay bound. Let us assume there is such a path p_x, and let r_r and h be the mrb and hop count of that path. Let us compare p_x with the path p_t in the entry with hop count h and bandwidth r_r. Since both paths have the same hop count and residual bandwidth, they will have identical values for the first two terms in Equation 1. However, since p_t has a lower delay, it will have smaller values for the last two terms than p_x, which means that p_t can meet a given delay bound by reserving less bandwidth r. This means that p_x must be equal to p_t.

To find a *widest-shortest path*, we can search the table column by column in increasing order of hop count, searching each column in decreasing order of bandwidth values. The first feasible path found is a widest-shortest path. Similarly, a *shortest-widest path* can be found by searching the table row by row.

The shortest-delay path and minimum-bandwidth path algorithms require that the entire table is constructed, *i.e.*, all iterations have to be executed. In contrast, the shortest-widest path and the widest-shortest path algorithms can be implement more efficiently by ordering the different bandwidth values in decreasing order. To select a shortest-widest path, the algorithm can terminate when the first feasible path is found. To select a widest-shortest path, in addition to ordering the residual bandwidth values, we can swap the loop order in lines (a) and (b) in Figure 1. With this slight modification, the first feasible path found is a widest-shortest path. Figure 3 gives pseudocode for the widest-shortest path algorithm.

4 Simulator Design

Our evaluation is based on a session-level event-driven simulator, which has been used in studying routing best-effort traffic [22] and traffic requiring bandwidth guarantees [20]. The simulator simulates the start and termination of sessions, and does route selection, admission control, and resource reservation. To increase the level of realism, the simulator does packet-level simulation of connection setup and tear-down, and of routing information distribution. For connection setup and tear-down this means that the latency of these operations is modeled. For routing information distribution, this means that we include both the effect of routing update overheads and of delayed state information on performance.

The simulator uses link-state routing, and routing information distribution is implemented using a simple reliable flooding protocol similar to that used in PNNI and OSPF. The default period is 30 seconds. It is easy to show that static routing, while simple, can give very poor performance for guaranteed traffic, so we focus on dynamic routing. As a result, the routing information includes both static topology information and dynamic residual bandwidth and buffer space. We assume on-demand routing with a routing cost of 10 msec, and connection set up and tear down costs of 3 msec/how and 1 msec/hop, respectively. Since the connection life times are fairly long, these costs have little impact on the results.

We used two topologies in our simulation: an MCI topology and a local area switched cluster topology (Figure 4). The link propagation delay is calculated according to the physical distance between nodes for the MCI topology.

4.1 Traffic Loads

Our traffic load consists of two classes of traffic: guaranteed sessions and the best-effort sessions, which are described below. Clearly, many different types of traffic loads could be considered. Our evaluation shows that low bandwidth connections (e.g. voice or small best-effort data transfers) have little impact on performance, so we focus on higher-bandwidth video connections and high bandwidth best-effort traffic.

The traffic entering the network is split between the two traffic classes according to a predetermined ratio. The traffic load can be either evenly or unevenly distributed. For evenly distributed load, a new session selects with equal probability any pair of switches as its source and destination. Despite

```
Widest_Shortest_Path(G : GRAPH, s : NODE, d : NODE, d : real, j : real, b : NODE → int)
    r_vec ← Sorting(L)                                   /* sort R_i (i ∈ L) in decreasing order, delete duplicates */
    E ← lenght of r_vec                                  /* the number of different link residual bandwidth */
    for v ∈ V do                                         /* for all nodes in G */
        N_v ← ⌊(b(v) − b)/L_max⌋                         /* hop count bound for node v by buffer space */
        max_hop = max(max_hop, N_v)
    m ← min(|V|, max_hop)                                /* |V| is the number of nodes in G */
    Initialize-Single-Source(G, s, E, r_vec)             /* initialization*/
    for h ← 1 to m − 1 do                                /* iteration over hop count */
        for k ← 1 to E do                                /* iteration over different values of link residual bandwidth */
            if (n_k ≥ h) then                            /* if h meets the hop count bound set by j */
                for (u, v) ∈ L do                        /* for all links in G */
                    Relax(u, v, k, l_k, h, N_v)
            if l_k(d) ≤ d then
                return π_k                               /* path found */
    return FALSE                                         /* no feasible path */

Initialize-Single-Source(G, s, E, r_vec)
    for k ← 1 to E do                                    /* iteration over different values of link residual bandwidth */
        n_k ← ⌊(r_vec[k] · j − b)/L_max⌋                 /* bound on hop count determined by j */
        for v ∈ V do                                     /* for all nodes in G */
            l_k[v] ← ∞                                   /* for r = r_vec[k] */
            π_k[v] ← NIL                                 /* initialize path */
    return

Relax(u, v, k, l_k, h, N_v)
    if N_v ≤ h and R_(u,v) ≥ r_vec[k] then               /* check hop count bound and link residual bandwidth */
        if l_k[v] > l_k[u] + l_k(u, v) then
            l_k[v] ← l_k[u] + l_k(u, v)
            π_k[v] ← u                                   /* update the path */
    return
```

Figure 3: Pseudo-code for the widest-shortest path algorithm

the network is designed to match the expected traffic conditions, the network load is often unevely distributed in the sense that more traffic between some source and destination pairs than other pairs. We simulate these scenarios of unevenly distributed load by designate a percentage of sessions to use a preselected set of source and destination pairs and the rest of the sessions randomly pick their source and destination.

4.2 Guaranteed Sessions

Sessions arrive according to Poisson arrival. The arrival rate is determined by the traffic load for all guaranteed sessions, the mean requested bandwidth, and the mean call holding time. For the leaky bucket $\langle \sigma, b \rangle$ specified by the traffic source, the token rate σ is uniformly distributed between $1 \sim 5$ megabits/second while the bucket size b is uniformly distributed over one of the two intervals: [4KB, 8KB] and [16KB, 20KB]. This models a combination of low and high quality video.

Most studies assume an exponential call holding time distribution recent studies [3] show that the call holding time distribution for conversations, facsimile, and voice mail connections has a large portion of very short calls and lognormal long-tail distributions. We follow the model suggested in [3]: most of our simulation use a holding time distribution that is a mixture of two normal distributions (F_1 and F_2) on a logarithmic time scale with the mixing probability α

$$F(x) = \alpha \cdot F_1(x) + (1 - \alpha) \cdot F_2(x)$$

Finally, the traffic source of a guaranteed session also needs to specify its delay bound. We assume that the end-to-end delay of a session is either uniformly distributed in the interval [80ms, 120ms] or in [200ms, 240ms]. Tighter end-to-end delay bounds require more bandwidth to be reserved. This allows us to explore the impact of delay bound distribution on the performance of routing algorithms.

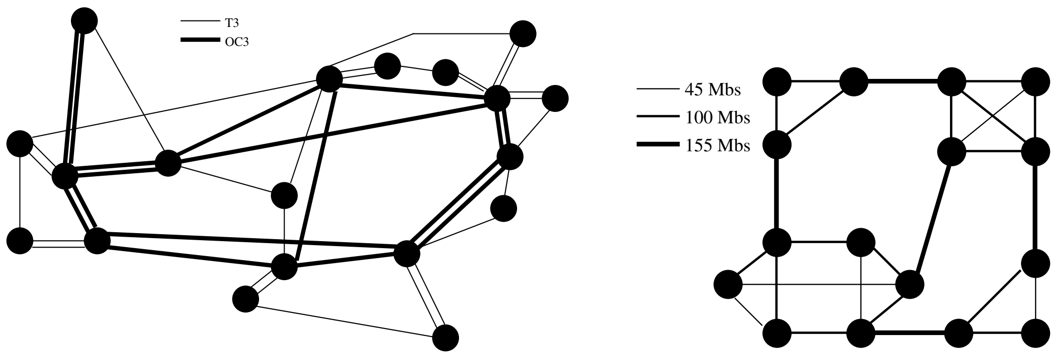
The performance metric for guaranteed traffic is the *call blocking rate*, the percentage of sessions being rejected by the network over the total number of arrival sessions:

$$\text{Call blocking rate} = \frac{\text{number of rejected guaranteed sessions}}{\text{number of arrival guaranteed sessions}}.$$

A guaranteed session can be rejected either because no path with sufficient resources can be found by the routing algorithm or because the resource availability on the selected path has changed since the time when the routing decision was made. We do not implement crankback.

4.3 Best-effort Sessions

We expect that in many networks, best-effort traffic will continue to be the most dominant traffic class. In that case, the call



(a) MCI topology (b) Switched cluster topology

Figure 4: Topologies used in the simulations

blocking rate can be a misleading performance metric since it does not capture how the routing algorithm for guaranteed traffic impacts the lower-priority best-effort traffic. For this reason we also look at the performance of best-effort traffic in networks that carry both guaranteed traffic.

Best-effort sessions specify how much data is to be sent, similar to [22]. The link capacity left over from guaranteed sessions is shared among all high-bandwidth sessions following the max-min fair share principle. This sharing model models ATM ABR service and under some conditions also IP networks. For example, it is shown in [14] that Round-Robin and window-based congestion control result in max-min fair sharing.

The performance metric for the best-effort sessions is the average throughput of the best-effort sessions, weighted by the size of the data transfer [17]:

$$\text{Average throughput} = \frac{\text{bytes sent by best}-\text{effort sessions}}{\text{time used by best}-\text{effort sessions}}.$$

5 Performance of Guaranteed Sessions

In this section, we examine the performance of the routing algorithms described in Section 3, assuming that the network carries only guaranteed traffic. We first consider scenarios when the network load is evenly distributed. We then consdier scenarios of unevenly distributed load.

5.1 Evenly Distributed Load

Figure 5 shows the call blocking rate as a function of the network load for the MCI topology for four different combinations of traffic burstiness and delay bounds.

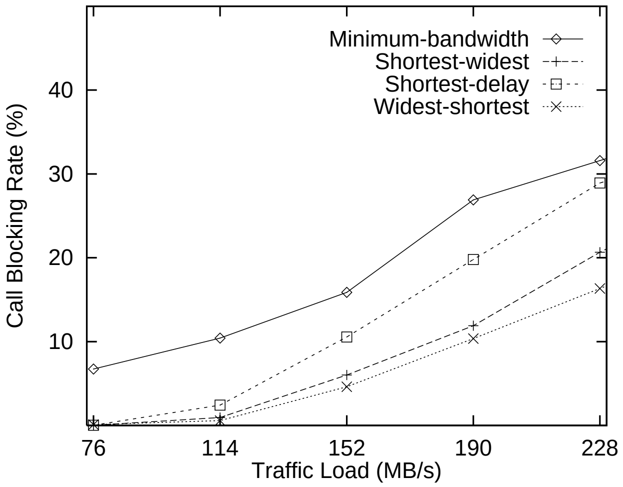
We see that the performance difference between the different routing algorithms can be large: up to a factor of two. We also see (comparing the top two figures and the bottom two graphs) that the call blocking rate is higher when the delay bound is tighter and when the traffic is more bursty. This is to be expected: more bandwidth has to be reserved to limit the queueing delay. When the load is light, all algorithms except the minimum-bandwidth path algorithm converge quickly and result in a zero call blocking rate.

Overall, the widest-shortest path performs better than the other three algorithms and the minimum-bandwidth path performs the worst. The reason is that the widest-shortest path algorithm selects paths with as few hops as possible to conserve resources. At the same time, it does load balancing by selecting the widest one among all paths with the same hop-count. On the other hand, the minimum-bandwidth path, which also tries to conserve resources, does not take into account the current load of the path being selected, and may pick a loaded path, thus blocking future arrivals. It is interesting to see that the shortest-widest path performs reasonably well and has a performance similar to the widest-shortest path when the delay bounds are between 200ms and 240ms. This suggests that the load is an important factor when selecting a path. The performance for the shortest-delay path is interesting. When the load is light, its performance converges to that of widest-shortest path. When the load is heavy, its performance moves close to that of minimum-bandwidth path. This can be explained using Equation (1). When the load is heavy, the r is small, which means that n in the equation weights much less than r in determining the final end-to-end delay. This results in selecting a path with higher r but with more hops.

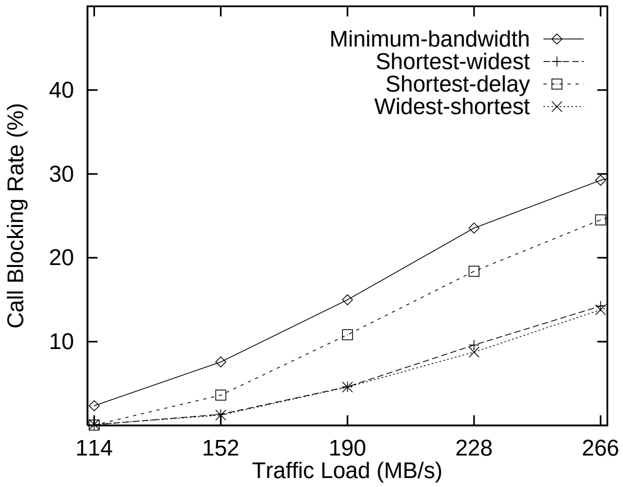
Figure 6 shows the call blocking rate for the switched cluster topology. We observe similar behavior for the different routing algorithms as for the MCI topology. The main difference is that the shortest-widest path does not perform as well as for the MCI topology: it often is worse than the minimum bandwidth path. Thus, the shortest-widest path is sensitive to the topology, which was also observed for best-effort traffic in [22].

5.2 Unevenly Distributed Load

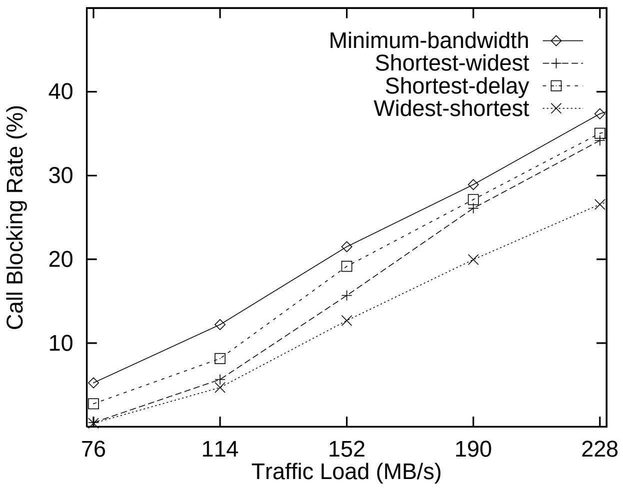
In Figure 7, the traffic load is unevenly distributed: 50% of the traffic is between the West coast and East coast and the



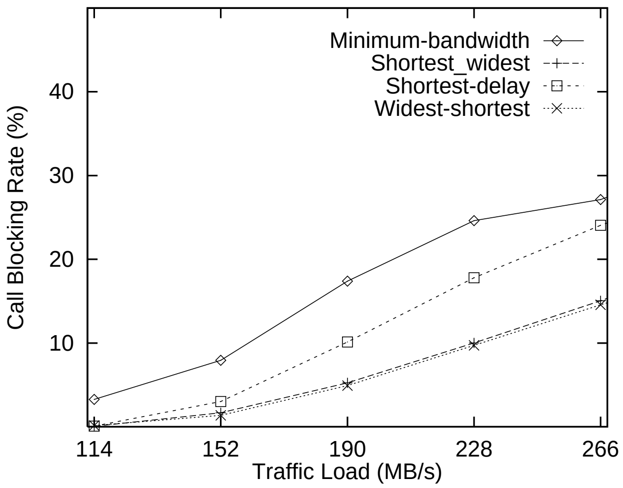
(a) b = [4KB, 8KB] and **d** = [80ms, 120ms]



(b) b = [4KB, 8KB] and **d** = [200ms, 240ms]

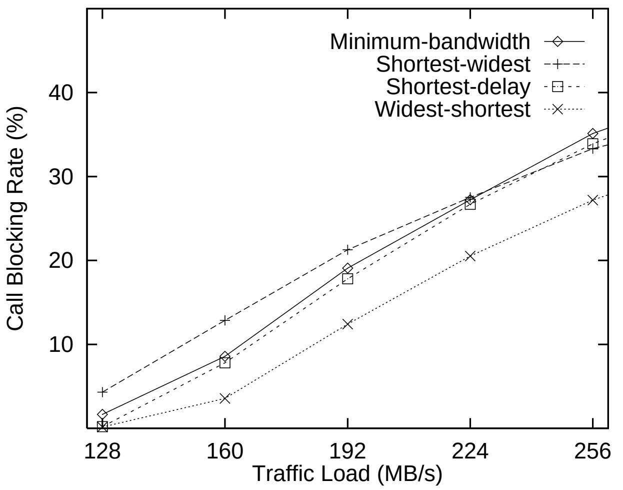


(c) b = [16KB, 20KB] and **d** =[80ms, 120ms]

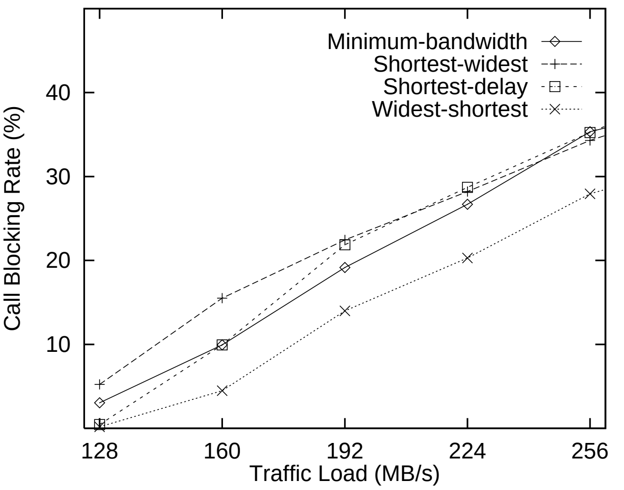


(d) b = [16KB, 20KB] and **d** =[200ms, 240ms]

Figure 5: Call blocking rate as a function of network load: MCI topology, 100% guaranteed sessions, and evenly distributed load



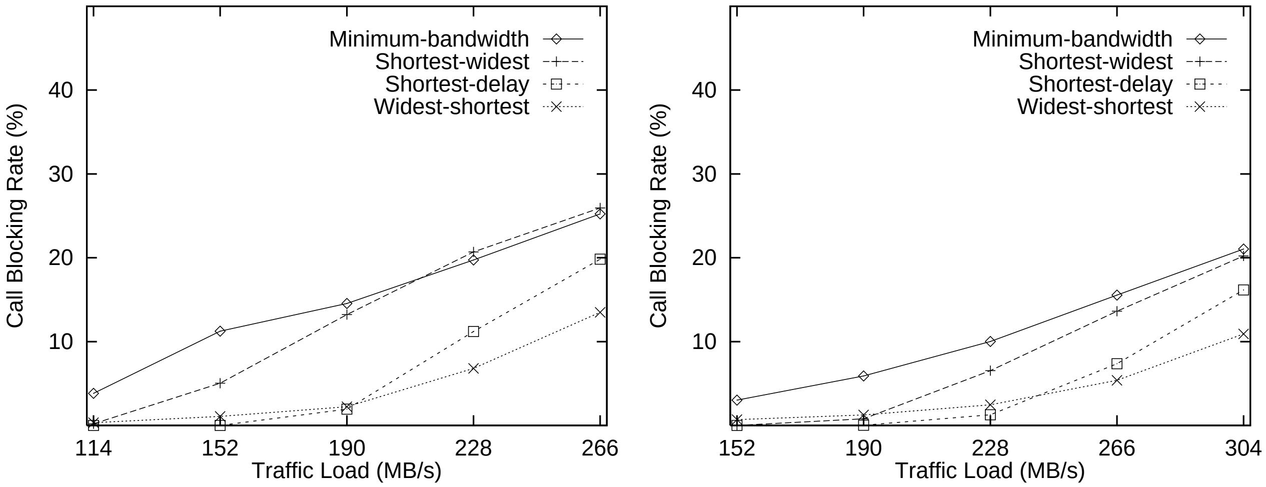
(a) b = [4KB, 8KB] and **d** = [200ms, 240ms]



(b) b = [16KB, 20KB] and **d** =[200ms, 240ms]

Figure 6: Call blocking rate as a function of network load: Switched cluster, 100% guaranteed sessions, and evenly distributed load

other 50% is evenly distributed. We again see a large performance difference between the different routing algorithms.



(a) b = [4KB, 8KB] and d = [80ms, 120ms]

(b) b = [16KB, 20KB] and **d** =[200ms, 240ms]

Figure 7: Call blocking rate as a function of network load: MCI topology, 100% guaranteed sessions, and unevenly distributed load

The widest-shortest path continues to perform well, although in when the load is very light, the shortest-delay path sometimes outperform the widest-shortest path. This suggests that using a path with the minimum hop-count may make it hard to avoid some congested links. It also indicates the importance of selecting an unloaded path rather than a minimum hop path when the load is relative light. The minimum-bandwidth path still performs badly as was the case for even traffic loads. The shortest-widest path keeps performing inconsistently.

In summary, for evenly distributed load, the widest-shortest path algorithm outperforms other three algorithms because of resource conservation. For unevenly distributed load, balancing load becomes more important. The shortest-delay path algorithm outperforms other algorithms when the load is light. However, when the load is heavy, conserving resources becomes more important, the widest-shortest path outperforms others. The minimum-bandwidth path algorithm performs poorly because it may select heavily loaded paths, while the shortest-widest path algorithm is sensitive to different topologies and load distribution.

The result on the widest-shortest path is different from what we observed for best-effort traffic [22], where the shortest-distance path algorithm has a clear performance edge over the other algorithms. This difference is caused by the different sharing policies of the two different traffic classes. Heavily loaded links become automatically ineligible for guaranteed sessions, causing any dynamic algorithm to route around them. But these links remain eligible for best-effort traffic so the algorithm has to explicitly avoid them.

6 Performance Impact on Best-effort Traffic

In a network with both guaranteed traffic and best-effort traffic, what path to use for guaranteed traffic can have a significant impact on the performance of best-effort traffic. Thus, it is important to understand this performance impact before adopting a routing algorithm. For example, when the load of guaranteed traffic is relatively low compared to the network capacity, the call blocking rate for guaranteed traffic is likely to be zero, regardless which routing algorithm is used. In such cases, the throughput for best-effort traffic is an important performance metric for distinguishing different routing algorithms that are used for guaranteed traffic.

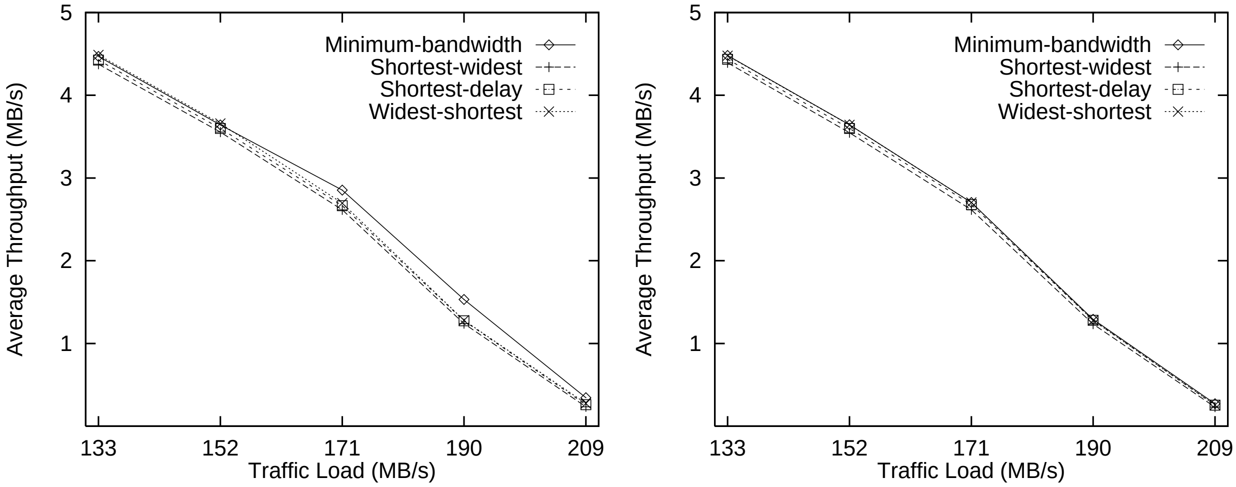
In this section, we examine this performance impact for networks supporting both guaranteed traffic and best-effort traffic. We focus on the throughput of best-effort sessions, assuming that all best-effort sessions share the unreserved link capacity according to the max-min fair sharing policy defined in [16] and the algorithm used for best-effort sessions is the shortest distance path with the distance for a path defined as

$$\text{dist}(P) = \sum_{i=1}^{k} \frac{1}{r_i}$$

where r_i is the max-min fair share rate a new best-effort session will obtain [22].

6.1 Evenly Distributed Load

Let us first look at networks with an evenly distributed traffic load. The total traffic load is split between guaranteed sessions and best-effort sessions, with 40% of the traffic for guaranteed sessions. Figure 8 shows the average throughput of the best-effort sessions as a function of the total network load. We assume that up to 90% of the capacity of each link can be reserved for guaranteed sessions. Overall, all four routing algorithms have a similar performance impact on best-effort sessions, although the minimum-bandwidth path performs slightly better than others. However, this slight performance advantage is achieved at the cost of more guaranteed sessions being blocked. Table 1 shows the call blocking rate



(a) b = [4KB, 8KB] and **d** = [80ms, 120ms]

(b) b = [16KB, 20KB] and **d** =[200ms, 240ms]

Figure 8: Average throughput as a function of network load: MCI topology, 40% guaranteed sessions, evenly distributed load, and 90% reservation rate

for guaranteed sessions. A much higher call blocking rate is observed when the minimum-bandwidth path is used, which is not surprising given the results in the previous section.

Figure 9 shows the performance impact for the switched cluster topology. For all traffic loads in the figure, no guaranteed sessions are blocked. What is surprising is that the shortest-widest path performs much worse than the three other algorithms. This is suggests that, for a more symmetric network topology, applying the shortest-widest path algorithms for guaranteed sessions can cause much higher resource consumption, therefore leaving fewer resources for best-effort sessions. This again demonstrates that the shortest-widest path is not performing consistently.

6.2 Unevenly Distributed Load

Under an unevenly distributed load, Figure 10 shows the impact of different routing algorithms on the performance of the best-effort sessions for the MCI topology, and Table 2 shows the call blocking rate for the guaranteed sessions. We see again that the shortest-widest path algorithm takes away more resource from best-effort sessions, especially when the load is heavy. The minimum-bandwidth path blocks more guaranteed sessions although its performance impact on best-effort sessions is close to the shortest-delay path and the widest-shortest path algorithms. Very low call blocking rates are observed for the widest-shortest path because of its limited ability to balance the network load. The shortest-delay path performs overall better: no guaranteed sessions are blocked and best-effort sessions achieve a high average throughput.

In summary, the shortest-widest path algorithm for guaranteed sessions consumes more resources, therefore results in low throughput for the best-effort sessions, The other three routing algorithms have similar performance impact on best-effort traffic.

7 Approximation Algorithms

In this section, we present approximation algorithms for the routing algorithms discussed in previous sections and evaluate their performance both in terms of running time and in terms of network throughput.

7.1 Algorithms

In the previous sections, we evaluated the performance of four different routing algorithms both in terms of the call blocking rate and in terms of the average throughput of the best-effort sessions. These algorithms have a running time $O(m \cdot e \cdot E)$, where m is the number of nodes, e the number of different values of link residual bandwidth, and E the number links in the network. In the worst case, e is equal to E, and the running time is $O(m \cdot E^2)$. We observe in our simulations that the link residual bandwidth that this worst case may be very common, since the residual bandwidth of two links will usually be different, although the difference may be small. The algorithms have to iterate over all these very close but different bandwidth values. We recall that the original motivation for iterating the Bellman-Ford algorithm over all possible values of the link residual bandwidth was to ensure that no path with the shortest delay is missed, regardless of the amount of bandwidth that has to be reserved.

A first optimization is to use a small set of residual bandwidths, and to approximate the residual bandwidth of each link by rounding down to one of these values. We can then iterate the Bellman-Ford algorithm over this smaller set of values, which enables us to eliminate iterations over close but different residual bandwidth values. A second optimization is to have an bound on the amount of bandwidth that a flow can reserve. In practice, there must be an up bound on the amount of bandwidth to be reserved for a given token rate. For example, if the

Traffic load (MB/s)	133	152	171	190	209
Minimum-bandwidth	0.06	0.13	0.08	1.44	3.68
Shortest-widest	0.00	0.00	0.00	0.01	0.07
Shortest-delay	0.00	0.00	0.00	0.01	0.22
Widest-shortest	0.02	0.03	0.03	0.04	0.13

(a) b = [4KB, 8KB] and **d** =[80ms, 120ms]

Traffic load (MB/s)	133	152	171	190	209
Minimum-bandwidth	0.00	0.00	0.01	0.08	0.78
Shortest-widest	0.00	0.00	0.00	0.00	0.01
Shortest-delay	0.00	0.00	0.00	0.00	0.01
Widest-shortest	0.00	0.00	0.00	0.00	0.04

(b) b = [16KB, 20KB] and **d** =[200ms, 240ms]

Table 1: Call blocking rate in percentage: MCI topology, 40% guaranteed sessions, evenly distributed load, and 90% reservation rate

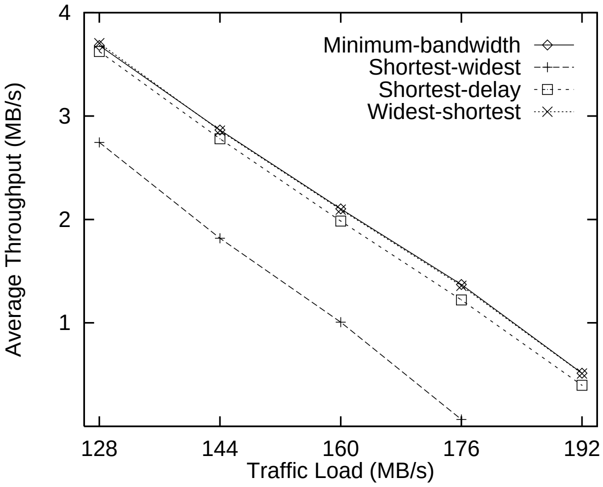


Figure 9: Average throughput as a function of network load: Switched cluster, 60% best-effort, 90% link-capacity reservation, evenly distributed load, b = [16KB, 20KB], and d =[200ms, 240ms]

token rate of a flow is 5 Mb/s, the network should not reserve more than 25 Mb/s of the link bandwidth in order to achieve a delay bound.

Our approximation algorithms select a set of bandwidth values $V = \{v_1, \ldots, v_n\}$ such that $v_1 > \ldots > v_n$. The value v_1 is the maximal bandwidth that can be reserved. For every link, we round its residual capacity v to the largest v_i such that $v_i \leq v$. The Bellman-Ford algorithm is iterated only over those v_i. At each iteration, it assumes that the value of v_i is the bandwidth to be reserved.

The selection of the set V depends on the link capacity and the distribution of bandwidth requests. In order to select as few values as possible without eliminating many potential feasible paths from consideration, a heuristic is to have a shorter distance between two consecutive values while these value decrease. In our simulation, we select the following 14 values:

$$V = \{24, 20, 16, 14, 12, 10, 8, 7, 6, 5, 4, 3, 2, 1\}$$

In this example, 24 is maximum bandwidth considered. Once the values of V are selected, the worst case running time of the iterative Bellman-Ford algorithm is $O(|V| \cdot m \cdot E)$, where $|V|$ is a constant. In other words, the approximation algorithms will have running times that differ from the Bellman-Ford algorithm times by up to a constant factor.

7.2 Performance on Network Throughput

The simulation results in Sections 5 and 6 show that the shortest-widest path and the minimum-bandwidth path perform inconsistently for different topologies because they put either too much weight on balancing the network load or on bandwidth consumption. Our evaluation of the approximation algorithm focuses on the shortest-delay path and the widest-shortest path.

Figures 11 and 12 show for the MCI topology the performance difference between the widest-shortest path and the

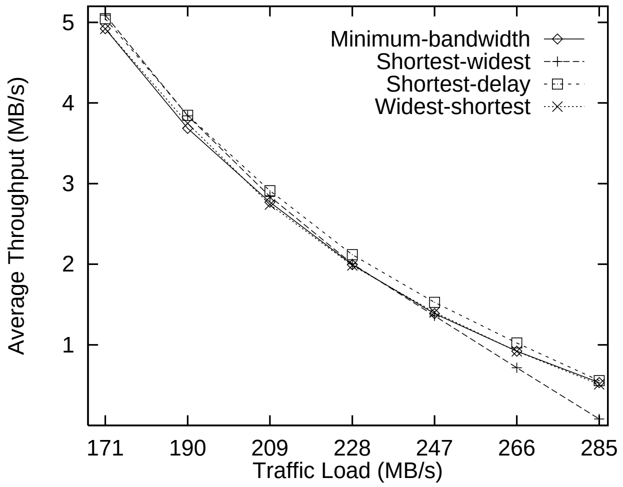


Figure 10: Average throughput as a function of network load: MCI topology, 40% guaranteed sessions, 90% link-capacity reservation, unevenly distributed load, b = [16KB, 20KB], and d =[200ms, 240ms]

Traffic load (MB/s)	209	228	247	266	285
Minimum-bandwidth	0.00	0.00	0.07	0.23	1.17
Shortest-widest	0.00	0.00	0.00	0.00	0.00
Shortest-delay	0.00	0.00	0.00	0.00	0.00
Widest-shortest	0.00	0.00	0.06	0.10	0.19

b = [16KB, 20KB] and **d** =[200ms, 240ms]

Table 2: Call blocking rate: MCI topology, 40% guaranteed sessions, unevenly distributed load, and 90% reservation rate

shortest-delay path algorithms, and their approximation, for even and uneven traffic loads respectively. We see that the call blocking rates for the widest-shortest path and its approximation are very similar, while the approximation algorithm for the shortest-delay path performs better than the original "correct" algorithm. This result is somewhat unexpected. Further analysis shows that the improvement is not caused by the limit on the reserved bandwidth, since the exact algorithms rarely allocate high bandwidth anyway. Instead the performance improvement is a result of the use of bandwidth intervals. Recall that the end-to-end delay (see Equation 1) is a function of the hop count and reservable bandwidth. When the load is heavy, the reservable bandwidth is small. Slightly reducing this value to its closest discrete interval value leads the shortest-delay path algorithm to select paths with fewer hops, therefore, to conserve network resources. However, the approximation scheme has little impact on the widest-shortest path algorithm, which is already very biased towards minimizing resource utilization. Figure 13 shows similar results for the switched cluster topology. Again, the approximation to the shortest-delay path algorithm performs better than the original shortest-delay path algorithm, although it does not perform as well as the widest-shortest path algorithm this time.

7.3 Performance on Running Time

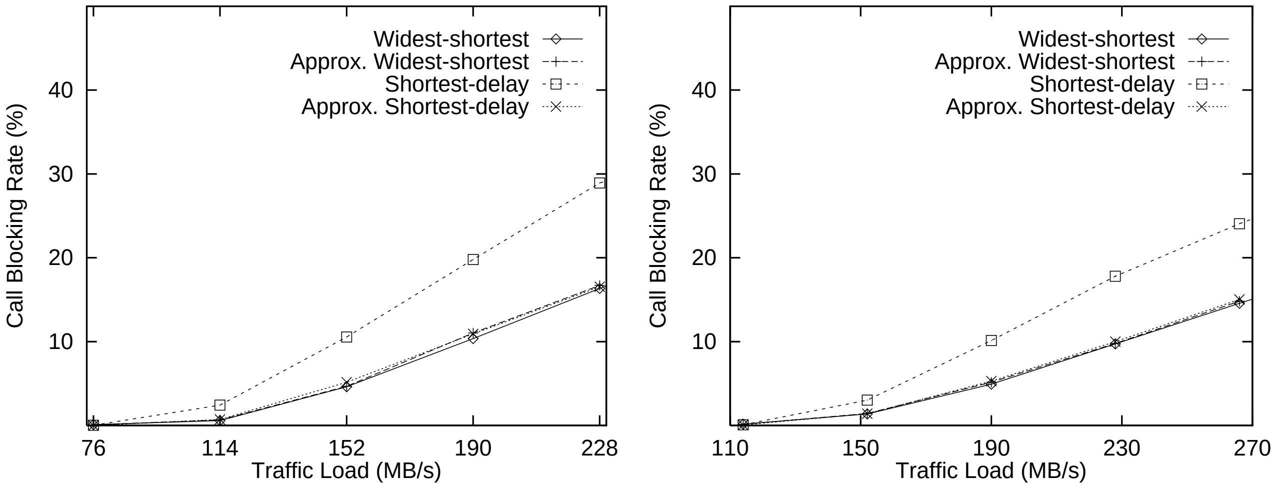
The running time we measured is the average algorithm running time in our simulation:

$$\text{Average running time} = \frac{\text{time used by routing algorithm}}{\text{number of flows}}.$$

We compare the average time of the approximation algorithms and the original iterative Bellman-Ford algorithms. The measurement was carried out on a DEC Alpha Station 255. Figure 14 shows the time (in milliseconds) needed for the widest-shortest path, the shortest-delay path, and their approximation. We see that the approximation algorithms run significantly faster than the original algorithms by up to a factor of 8. Moreover, we see that the widest-shortest path algorithm and its approximation run 18% faster than the shortest-delay path and its approximation, respectively. The significant improvement on the running time of our approximation algorithms comes from eliminating iterations over similar but different values of the link residual bandwidth.

8 Related Work

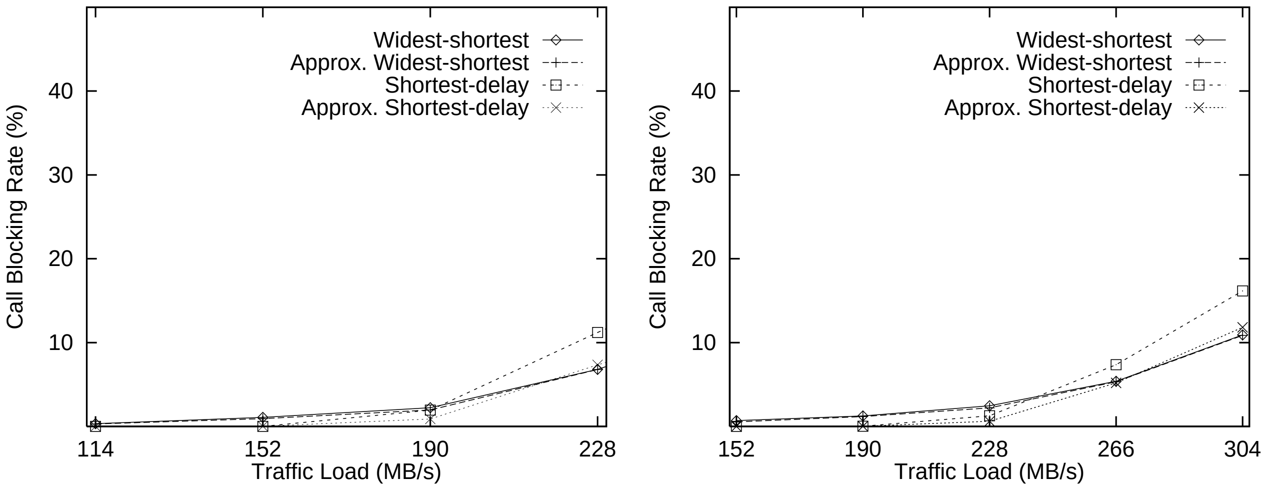
Many studies in the literature have addressed different aspects of QoS routing. A good introduction to QoS routing and existing routing techniques can be found in [18, 30]. Trunk Reservation and Dynamic Alternative Routing [8], and Least



(a) b = [4KB, 8KB] & d = [80ms, 120ms]

(b) b = [16KB, 20KB] & d = [200ms, 240ms]

Figure 11: Call blocking rate as a function of network load: MCI topology, 100% guaranteed sessions, and evenly distributed load



(a) b = [4KB, 8KB] & d = [80ms, 120ms]

(b) b = [16KB, 20KB] & d = [200ms, 240ms]

Figure 12: Call blocking rate as a function of network load: MCI topology, 100% guaranteed sessions, and unevenly distributed load

Load Routing [13] are some of the algorithms that have been studied in telecommunications networks. In [5], a QoS based routing framework is defined and many challenging questions are raised.

Several studies have investigated routing algorithmic problems associated with QoS routing. In [7, 33], it is shown that the QoS routing problem of finding a path satisfying both delay and delay-jitter constraints is NP-complete. Jaffe and Salama studied [15, 28] heuristics to tackle the NP-problem of routing with two additive constraints. Przygienda suggested in [25] the use of cost vectors to solve the NP-complete by associating a priority to each constraint. Rampal [26] evaluated the performance of several path selection algorithms under the assumption of three different scheduling algorithms. The IBF algorithm that is used in this paper was first described in [21]. Zhao and Tripathi in [37] proposed a similar algorithm, but issues related to selecting efficient paths and resource utilization efficiency are not addressed. Several other studies have presented routing algorithms that are similar to or special cases of ours but in different context. In [27], routing algorithms for achieving the shortest delay when moving a burst of bytes are proposed. In [24], QoS routing algorithms are studied for finding the shortest-delay path when the network employs the weighted fair queueing service discipline. In [11], QoS routing algorithms for a network with inaccurate link-state information are explored. Guerin, Orda, and Williams outlined in [12] how OSPF can be extended to QoS routing, using the widest-shortest feasible paths; no performance evaluation is provided. QoS routing algorithms for traffic requiring bandwidth guarantees are addressed in [20] and related work can also be found there.

In contrast to the previous studies, we presented not only

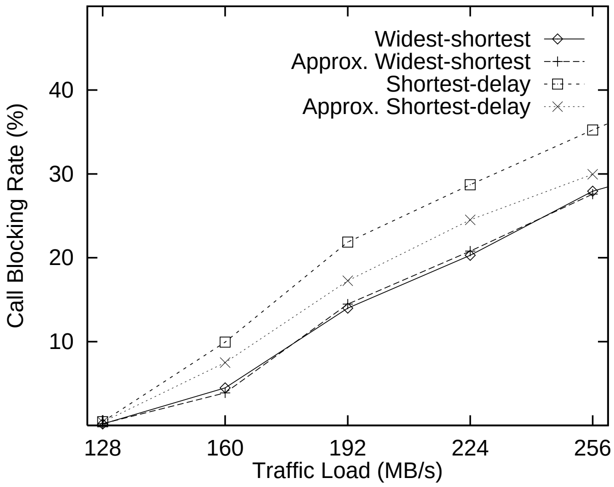


Figure 13: Call blocking rate as a function of network load: Switched cluster, 100% video traffic, evenly distributed load, b = [16KB, 20KB], and d = [200ms, 240ms]

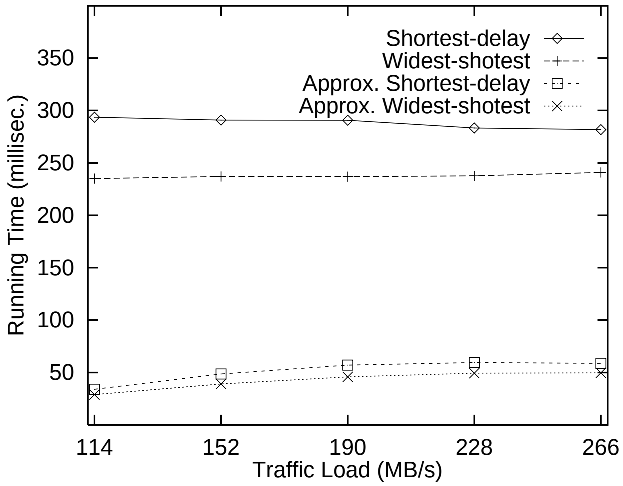


Figure 14: Average running time as a function of network load: MCI topology, 100% guaranteed sessions, 90% link-capacity reservation, unevenly distributed load, b = [16KB, 20KB], and d =[200ms, 240ms]

a complete set of QoS routing algorithms for traffic with delay, delay jitter, and buffer space constraints with a variety of optimality criteria, but also approximation algorithms that run significantly faster than the original algorithms without sacrificing resource utilization efficiency. We also present a detailed performance evaluation for four different algorithms, considering both call blocking rate and impact on best-effort traffic as performance metrics.

9 Conclusion

In order to support applications with stringent QoS requirements such as interactive continuous media applications, both the ATM and IP community have defined "guaranteed" service classes that provide per-flow guarantees for delay, delay-jitter, and loss rate. QoS routing is one of important resource management components that support QoS in integrated services networks.

In this paper we address two important issues related to achieving resource efficiency while selecting feasible paths and the effectiveness and practicality of routing algorithms. To achieve low blocking rates, we identified four optimization criteria that place a different emphasis on minimizing the resource utilization and load distribution in the network. Based on the IBF algorithm, we developed path selection algorithms that select paths with these different criteria. The performance of of these different routing algorithms is evaluated through an extensive simulation study, using realistic topology, traffic model, and periodic routing information distributions. Unlike telecommunication networks, multiple classes of service will be supported in data networks. Our performance evaluation considers not only the blocking rate for guaranteed traffic but also the influence on the throughput of best-effort traffic. Our results show that the performance gap for these different algorithms is large and both conserving resources and balancing the network load are necessary to achieve high network throughput. The widest-shortest path algorithm performs well for heavy load because it conserves resources, but the shortest-delay path has a performance edge for light load because it balances the load. Surprisingly, the minimum-bandwidth path performs worst because it may distribute load unevenly over different links.

Even though the IBF algorithms are polynomial, their computational complexity is significantly higher than the Bellman-Ford algorithm. We develop an approximation to the iterative Bellman-Ford algorithm that runs almost ten times faster than the exact algorithm. The simulation results show that there is no loss in routing accuracy and the overall network performance. Overall, the approximation algorithm for the shortest-delay path performs consistently well.

Our results demonstrate that QoS routing is desirable and feasible. It is desirable because of the significant throughput improvement by using carefully selected routes, and it is feasible because QoS routing algorithms run as fast as the Bellman-Ford algorithm up to a constant factor. The routing information needed is link residual bandwidth and link propagation delay.

References

[1] ATM Forum Traffic Management Specification Version 4.0, October 1995. ATM Forum/95-0013R8.

[2] J.C.R. Bennett and H. Zhang. WF^2Q: Worst-case Fair Weighted Fair Queueing. In *Proceedings of IEEE INFOCOM'96*, May 1996.

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

[4] David Clark, Scott Shenker, and Lixia Zhang. Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism. *ACM SIGCOMM 92*, August 1992.

[5] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick. A Framework for QoS-based Routing in the Internet. *IETF Internet Draft <draft-ietf-qosr-framework-01.txt>*, July 1997.

[6] A. Demers, S. Keshav, and S. Shenker. Analysis and Simulation of a Fair Queueing Algorithm. *ACM SIGCOMM 89*, 19(4):2–12, August 19-22, 1989.

[7] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979. (page 214).

[8] R.J. Gibbens, F.P. Kelley, and P.B. Key. Dynamic Alternative Routing — Modelling and Behaviour. In *Proceedings of the 12th International Teletraffic Congress*, June 1988.

[9] S. Golestani. A Self-Clocked Fair Queueing Scheme for Broadband Applications. In *Proceedings of IEEE INFOCOM'94*, pages 636–646, Toronto, Canada, June 1994.

[10] P. Goyal and H.M. Vin. Generalized Guaranteed Rate Scheduling Algorithms: A Framework. Technical Report Technical Report TR95-30, Department of Computer Science, UT Austin, Texas, 1995.

[11] R. Guerin and A. Orda. QoS-based Routing in Networks with Inaccurate Information: Theory and Algorithms. In *Proc. IEEE INFOCOM'97*, 1997.

[12] R. Guerin, A. Orda, and D. Williams. QoS Routing Mechanisms and OSPF Extensions. *IETF Internet Draft <draft-guerin-qos-routing-ospf-00.txt>*, November 1996.

[13] S. Gupta, K.W. Ross, and M. El Zarki. On Routing in ATM Networks. In M. Steenstrup, editor, *Routing in Communications Networks*, pages 49–74. Prentice Hall, 1995.

[14] E. Hahne. Round-Robin Scheduling for MaxMin Fairness in Data Networks. *IEEE Journal on Selected Areas in Communications*, 9(7), September 1991.

[15] J. Jaffe. Algorithms for Finding Paths with Multiple Constraints. *Networks*, 14(1):95–116, Spring 1984.

[16] J.M. Jaffe. Bottleneck flow control. *IEEE Transactions on Communications*, COM-29(7):954–962, July 1981. Correspondence.

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

[18] W.C. Lee, M.G. Hluchyj, and P.A. Humblet. Routing Subject to Quality of Service Constraints in Integrated Communication Networks. *IEEE Network*, 9(4):14–16, July/August, 1995.

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

[20] Q. Ma and P. Steenkiste. On Path Selection for Traffic with Bandwidth Guarantees. In *IEEE International Conference on Network Protocols*, Atlanta, Georgia, October 1997.

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

[22] Q. Ma, P. Steenkiste, and H. Zhang. Routing High-Bandwidth Traffic in Max-Min Fair Share Networks. In *ACM SIGCOMM '96*, pages 206–217, Stanford, CA, August 1996.

[23] A. Parekh and R. Gallager. A Generalized Processor Sharing Approach to Flow Control-the Single Node Case. *IEEE/ACM Transactions on Networking*, 3(1):344–357, June, 1993.

[24] C. Pornavalai, N. Shiratori, and G. Chakraborty. QoS Based Routing Algorithm in Integrated Services Packet Networks. In *IEEE International Conference on Network Protocols*, Atlanta, Georgia, October 1997.

[25] A. B. Przygienda. *Link State Routing with QoS in ATM LANs*. PhD thesis, Dipl. Inf. Ing. ETH, 1995.

[26] S. Rampal. *Routing and End-toend Quality of Service in Multimedia Networks*. PhD thesis, North Carolina State University, August, 1995.

[27] J.B. Rosen, S.Z. Sun, and G.L. Xue. Algorithms for the Quickest Path Problem. *Computers and Operations Research*, 18(6):579–584, 1991.

[28] H.F. Salama, D.S. Reeves, and Y. Viniotis. A Distributed Algorithm for Delay-Constrained Routing. In *Proceedings of IEEE INFOCOM'97*, Kobe, Japan, April 1997.

[29] S. Shenker, C. Partridge, and R. Guerin. Specification of Guaranteed Quality of Service. *IETF Internet Draft <draft-ietf-intserv-guaranteed-svc-07.txt>*, February 1997.

[30] M. Steenstrup. *Routing In Communications Networks*. Prentice Hall, Englewood Cliffs, NJ 07362, 1995.

[31] D. Stiliadis and A. Varma. Frame-based Fair Queueing: A New Traffic Scheduling Algorithm for Packet-Switched Networks. In *ACM SIGMETRICS 96*, Philadelphia, PA, May 1996.

[32] D. Stiliadis and A. Varma. A General Methodology for Designing Efficient Traffic Scheduling and Shaping Algorithms. In *Proceedings of IEEE INFOCOM'97*, Kobe, Japan, April 1997.

[33] Z. Wang and J. Crowcroft. Quality-of-Service Routing for Supporting Multimedia Applications. *IEEE JSAC*, 14(7):1288–1234, September 1996.

[34] H. Zhang. Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks. *Proceedings of the IEEE*, 83(10), October 1995.

[35] L. Zhang. Virtual Clock: A New Traffic Control Algorithm for Packet Switching Networks. *SIGCOMM 90*, pages 19–29, 1990.

[36] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A New Resource Reservation Protocol. *IEEE Communications Magazine*, 31(9):8–18, September 1993.

[37] W. Zhao and S. Tripathi. Routing Guaranteed Quality of Service Connections in Integrated Services Packet Networks. In *IEEE International Conference on Network Protocols*, Atlanta, Georgia, October 1997.