Routing High-bandwidth Traffic in Max-min Fair Share Networks

Qingming Ma Peter Steenkiste Hui Zhang
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
{qma, prs, hzhang}@cs.cmu.edu

Abstract

We study how to improve the throughput of high-bandwidth traffic such as large file transfers in a network where resources are fairly shared among connections. While it is possible to devise priority or reservation-based schemes that give high-bandwidth traffic preferential treatment at the expense of other connections, we focus on the use of routing algorithms that improve resource allocation while maintaining max-min fair share semantics. In our approach, routing is closely coupled with congestion control in the sense that congestion information, such as the rates allocated to existing connections, is used by the routing algorithm. To reduce the amount of routing information that must be distributed, an abstraction of the congestion information is introduced. Using an extensive set of simulation, we identify a link-cost or cost metric for "shortest-path" routing that performs uniformly better than the minimal-hop routing and shortest-widest path routing algorithms. To further improve throughput without reducing the fair share of single-path connections, we propose a novel prioritized multi-path routing algorithm in which low priority paths share the bandwidth left unused by higher priority paths. This leads to a conservative extension of max-min fairness called prioritized multi-level max-min fairness. Simulation results confirm the advantages of our multi-path routing algorithm.

1 Introduction

Future integrated-service networks will support a wide range of applications with diverse traffic characteristics and performance requirements. To balance the need for efficient resource management within the network against the diversity of applications, networks should provide multiple classes of services. While most service models recognize only a single best-effort traffic class, the growing diversity of applications can be classified into at least three traffic types. First, **low-latency traffic** consists of small messages, each sent as a single packet a small number of packets, so the key performance index is the end-to-end per packet delay; a typical example is RPC. Second, **continuous-rate traffic** consists of a continuous traffic stream with a certain intrinsic rate, i.e. the application does not benefit from bandwidths higher than the intrinsic rate; Internet video applications such as nv and vic are examples. Finally, **high-bandwidth traffic** consists of transfers of large blocks of data; examples are Web browsing and I/O intensive scientific computations. The key performance index is the elapsed time, which is determined by average throughput rather than packet delay. In contrast to the other two traffic classes, it can typically consume as much network bandwidth as is available. Given the large diversity in best-effort applications, we argue that they should not all be handled in the same way by the network.

For best effort traffic, network resources are shared dynamically by all applications. Resources are usually allocated by two mechanisms operating on different time scales. At a coarser time scale, a *routing* entity directs traffic along less congested paths to balance the network load. At a finer time scale, *congestion control* mechanisms dynamically adjust the source transmission rate to match the bandwidth available on the chosen path. Since traditional data networks use a connectionless architecture with no per-session state inside the network, routing is usually performed on a per-packet basis and congestion control is performed end-to-end with no network support. The trend in high speed networks is to have a connection-oriented architecture with per-session state inside the network. Congestion control algorithms that exploit the per-session state have been studied widely, and this has resulted in algorithms that provide max-min fair sharing between competing applications. However, routing algorithms for this new environment have been largely neglected.

In this paper, we study how we can use routing to improve the performance of high bandwidth applications in networks that employ max-min fair share based congestion control. The routing entity makes use of rate information generated by the traffic management algorithm. Linking the two resource allocation mechanisms makes it possible to do effective load-sensitive routing. We propose an abstraction of max-min rate information that can be used efficiently as the routing link-state and we identify a "link cost" that is suitable for routing high-bandwidth traffic. The dynamic, complex nature of resource sharing in max-min fair sharing networks makes this a difficult task. Our evaluation is based on simulation of several traffic loads on a number of network topologies. We study both single-path and multi-path algorithms. For multi-path routing, we introduce a prioritized multi-level max-min fairness model, that allows low priority paths to share bandwidth left unused by higher priority paths. This approach preserves the original single path max-min fair share semantics and prevents a multi-path connection from grabbing an unfair amount of bandwidth by using a large number of paths.

The remainder of this paper is organized as follows. In Section 2 we present our general approach. We describe the routing algorithms and discuss their implementation issues in Section 3. Section 4 describes our evaluation methodology

and Sections 5 through 7 present our results. In Section 8 we discuss related work and we summarize in Section 9.

2 Problem statement

In this section we review the characteristics of high-bandwidth applications and max-min fair share networks and we present our approach to routing.

2.1 High bandwidth traffic

The main performance index for a high-bandwidth traffic stream is the **elapsed time**, i.e. the period from when the application issues the transfer command to when the transfer finishes. The elapsed time E_i for traffic stream i consists of the following terms:

$$E_i = P_i + D_i + \frac{b_i}{r_i} \tag{1}$$

where P_i is the connection establishment time, D_i is the end-to-end packet delay, b_i is the size of the data transfer, and r_i is the average rate.

For high-bandwidth applications, the message size b_i is very large, so the elapsed time E_i is dominated by the last term. Since b_i is a constant, we minimize E_i by maximizing the average rate r_i. One way of increasing r_i is to give session i a higher priority, larger service share, or reserved bandwidth. All these mechanisms give session i preferential treatment at the expense of other sessions, and they require external administrative or pricing policies to function properly. In contrast, we focus on max-min fair networks, where all traffic streams are treated equally by the traffic management algorithm. It is still possible to enhance the performance of high-bandwidth traffic streams by routing them along paths that will yield, on average, higher rates. This requires a routing algorithm that can estimate the expected rate for new connections along any paths.

2.2 Max-min fair share network

The concept of max-min fair share was first proposed by Jaffe to address the issue of fair allocation of resources in networks based on virtual circuits (VC) or connections [14]. Recently, it has received much attention [6, 4] because it has been identified by the ATM Forum as one of main design goals for ABR traffic management algorithms.

Intuitively, if there are N connections sharing a link of a max-min fair share network, each will get one "fair share" of the link bandwidth. If a connection cannot use up its fair share bandwidth because it has a lower source rate or it is assigned a lower bandwidth on another link, the excess bandwidth is split "fairly" among all other connections. There are several definitions of "fair share". In the basic definition, each of the N connections competing for the link or excess bandwidth, gets one N^{th} of the bandwidth. Other definitions, such as weighted fair sharing and multi-class fair sharing, require pricing or administrative policies to assign a weight or class for each connection. Since these policies are still an active area of research, we expect the basic max-min fair share model to be the first one to be supported by commercial ATM networks, and we will use it in this paper.

There are two ways of implementing a max-min fair share network. One option is that all switches use a Fair Queueing [9] scheduler. The other option is to have switches explicitly compute the *max-min fair rate* for each connection and inform each source of this rate; sources are required to send no more than their max-min fair rate [6, 7]. While the first approach requires per-connection queueing in the switch, the second one does not, thus simplifying switch design. This is one of the reasons why the ATM Forum selected the implementation based on explicit rate calculation. This approach also has the advantage that switches have rate information for all connections, and we will make use of this information to do load sensitive routing.

We now describe a simple centralized algorithm to calculate the max-min fair rates or saturation rates of all connections. A connection is called saturated if it has reached its desired source rate or a link on the path traversed by the connection is saturated. A link is called saturated if all of its bandwidth has been allocated to connections sharing the link. Let CN be the set of all connections in the network, CN_l the set of connections using l, and sat and unsat the set of saturated and unsaturated connections, respectively. Let sat_l be the set $\mathsf{CN}_l \cap \mathsf{sat}$, and unsat_l the set $\mathsf{CN}_l \cap \mathsf{unsat}$. Given a set S of connections, let $\mathsf{L(S)}$ be the set of all links in the network with at least one connection in S using them. Let C_l be the capacity of the link l. The algorithms can be described as follows:

1. Initialization: $\mathsf{sat} = \emptyset$, and $\mathsf{unsat} = \mathsf{CN}$.
2. Iteration: Repeat the following steps until unsat becomes $\emptyset$
   - For every link $l \in \mathsf{L} = \mathsf{L(unsat)}$, calculate
   $$\mathrm{inc}_l = \frac{C_l - \sum_{i \in \mathsf{sat}_l} r_i}{|\mathsf{unsat}_l|} \tag{2}$$
   - Get the minimum: $\mathrm{min_inc} = \min\{\mathrm{inc}_l \mid l \in \mathsf{L}\}$.
   - Update rate r_i: $r_i = r_i + \mathrm{min_inc}$.
   - Move new saturated connections in unsat to sat.

Note that the max-min fair rate of a connection is a function of time — it can change when new connections arrive or depart.

2.3 Routing approach

We will use link-state source routing algorithms. Link-state routing means that each node knows the network topology and the cost associated with each link [23, 3]. Source routing means that the source selects the entire path. Link-state source routing algorithms are particularly suitable for load-sensitive routing [5] and make it possible to select paths on a per-connection basis. Note that the ATM Forum also adopted hierarchical link-state source routing [10].

Link-state routing algorithms are typically based on Dijkstra's shortest path algorithm [11]; algorithms differ in the function that is used as the link cost. Since we are focusing on high-bandwidth traffic, we will use the (expected) fair share bandwidth for the connection in calculating the link cost. The fair share bandwidth of a new connection is estimated based on the rate information available in the switches, as we describe in Section 3.

Using the max-min fair rate as a cost function is unique. The routing algorithm used in most networks tries to minimize the number of hops (link cost is 1), or, for load sensitive routing, the end-to-end delay (link cost is packet delay). Neither cost function is necessarily a good predictor of available bandwidth. A common link cost function in reservation-based networks is the residual link bandwidth —

unreserved bandwidth. We cannot use residual bandwidth since the nature of bandwidth sharing in reservation-based and max-min fair share networks is very different. In contrast, an estimate of max-min fair rate that accounts for the nature of bandwidth sharing is an accurate load-sensitive predictor of the bandwidth available to a new connection.

3 Routing algorithms for high bandwidth traffic

We describe our single-path and multipath routing algorithms, and our approach to max-min fair rate estimation.

3.1 Single path: link cost functions

When trying to maximize bandwidth, it seems natural to pick the "widest path" algorithm, i.e. to select the path with the highest current max-min fair rate. This is however not necessarily the best link cost. The key observation is that the max-min fair rate changes over time, and the high fair rate available at connection establishment time may not be sustained throughout its lifetime of the connection. Since the fair rate of a connection is the minimum of the rates available on each link, the chance that the fair rate will go down increases with the number of hops. Moreover, longer paths consume more resources, which may reduce the rate of future connections. In summary, a "good" link cost has to balance the effects of the path length and the current max-min fair rate.

Toward this goal, we define a family of polynomial link costs, $(\frac{1}{r})^n$, where r is the current max-min rate for a new connection (see [16] for the use of polynomial costs in solving optimal graph cut problem). By changing n, we can cover the spectrum between shortest ($n = 0$) and widest ($n \to \infty$) path algorithms. In the remainder of this paper we will consider the following five algorithms:

- **Widest-shortest path**: a path with the minimum number of hops. If there are several such paths, the one with the maximum max-min fair rate is selected.
- **Shortest-widest path**: a path with the maximum max-min rate. If there are several such paths, the one with the fewest hops is selected.
- **Shortest**-dist(P, n): a path with the shortest distance

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

  where $r_1, \cdots, r_k$ are the max-min fair rates of links on the path P with k hops. We will consider three cases corresponding to $n = 0.5, 1$, and 2.

An interesting point is that dist($P, 1$) can be interpreted as the bit transmission delay from the traffic source to the destination should the connection get the rate r_i at hop i (Note: $min_i\{r_i\}$ is the connection's max-min fair share rate along the path). This delay is different from the measured delay used in traditional shortest delay paths in two ways. First, the focus is on bit transmission delay (i.e. bandwidth) instead of total packet delay. Second, the measure is for data belonging to a specific connection instead of a link average.

3.2 Link-state representation

To use the link cost functions defined above, we need to know the expected max-min rate r_i for a new connection at each link. The calculation of r_i requires to access rate information of all connections associated with this link. Since the number of connections can be very large, the volume of rate information can be large as well. To address this problem, we introduce a concise data structure to represent this rate information, and propose an approximate algorithm to calculate r_i. Instead of having a separate rate entry for each connection, we use a fixed number of discrete rate intervals. The rate information is simply represented by the number of connections with max-min rate in each interval. The size of this representation scales with the number of rate intervals, but it is independent of, and therefore scales well with, the number of connections sharing the link.

For each interval i, let `rate_scal[i]` be the middle value of the interval, and `num_conn[i]` the number of connections in the interval. For example, for a link of 155 Mb/s, rate information of a link can be represented by a vector of 64 entries with a scale function defined by

$$\texttt{rate_scal}(i) = \begin{cases} 0.5 * i & \text{if } 0 \le i < 16 \\ 1.0 * i - 8 & \text{if } 16 \le i < 32 \\ 1.5 * i - 24 & \text{if } 32 \le i < 48 \\ 2.0 * i - 48 & \text{if } 48 \le i < 64 \end{cases}$$

We used multiple scales in this example to ensure the accuracy for connections with low rates while limiting the size of the vector. The max-min rate for a new connection can now be estimated by

```
int i = 0, num_below_ave = 0;
int N = num_conn_using_the_link + 1;
float rate_below_ave = 0.0;
do {
    rate = (C - rate_below_ave) / (N - num_below_ave);
    tmp_num_below_ave = 0;
    while (rate_scale(i) < rate) {
        rate_below_ave += num_conn[i] * rate_scale(i);
        tmp_num_below_ave += num_conn[i];
        i++;
    }
    num_below_ave += tmp_num_below_ave;
} while (tmp_num_below_ave > 0)
return rate;
```

It is important to realize that both the rate representation and algorithm to calculate max-min fair rate are approximations. However, this is sufficient since the max-min fair rate will change over time and we are only interested in maximizing the max-min fair rate averaged over the connection's life time.

3.3 Multi-path: prioritized multi-level max-min fairness

One technique to increase the average throughput of a high-bandwidth traffic session is to use multiple parallel paths to transfer the data. Each path is realized using a single network-level connection. However, simply having high bandwidth applications use multiple paths is not acceptable since "max-min fairness" is implemented on the basis of network-level connections and a session with multiple paths (i.e. network-level connections) will receive a higher performance *at the expense of* the performance of sessions using only one path. Actually, applications could increase their bandwidth almost arbitrarily by using more paths.

To take advantage of higher throughput offered by multiple paths, without violating the fairness property, we propose a prioritized multi-level max-min fair share model. In

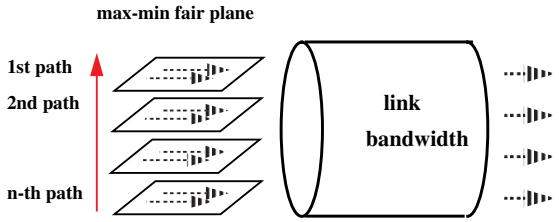


Figure 1: Prioritized multi-level max-min fair share

this model, connections are assigned to different priorities. If a session has N paths, the n^{th} path, $n = 1, \cdots, N$, is assigned to the n^{th} priority level. The number of priority levels in the network determines the maximum number of parallel paths (or connections) one session can have. For a network with M priority levels (Figure 1), M rounds of max-min fair share rate computations are performed. In the m^{th} round, the algorithm computes the max-min fair share rate for all the connections in level m using the residual link bandwidth left unused by higher priority connections, that is,

$$\text{inc}_l = \frac{(C_l - \sum_{k<m} R^k) - \sum_{i \in \mathsf{sat}_l^m} r_i^m}{|\mathsf{unsat}_l^m|} \qquad (3)$$

where R_k is the total max-min fair rate of all connections with the priority k.

This model has two interesting features. First, the multi-level fair share model is consistent with the single-level fair share model in the sense that the max-min fair rate for sessions using one path will not be reduced by the presence of multi-path sessions. Second, the priority levels are used in the max-min rate computation, but they do not necessarily have to be directly supported or even be visible by schedulers on switches and sources. For example, the rate-based congestion control adopted by ATM forum can easily be extended to prioritized multi-level fair sharing. The changes needed are the assignment of a priority to each connection and the use of a different algorithm for the fair rate calculation. The schedulers on the sources do not have to be changed: they continue to enforce the explicit rate assigned to them by the network. Similarly, switches can continue to use FIFO scheduling.

The multi-path routing algorithm based on prioritized multi-level fair sharing simply repeat a single-path routing algorithm at each level of max-min fair sharing, starting with the highest priority. At each level, the algorithm only uses bandwidth left unused by paths in the higher priority level. Note that this means that links that are saturated at a certain level will not be present in the network topology used at lower levels. The algorithm terminates either paths with sufficient bandwidth have been found or no more new paths with nonzero bandwidth can be found.

Finally, striping data over parallel paths is likely to introduce some overhead on the sending and receiving host, for example to deal with out of order packet arrival. Multi-path routing should therefore be used only if the expect increase in bandwidth is above a certain threshold.

4 Simulator design

We briefly describe our simulator and the simulation parameters that were used to collect results.

4.1 Simulator

We designed and implemented a session-level event-driven simulator. The simulator allows us to specify a topology and code modules representing traffic sources, the algorithm used by the switches for distributing bandwidth between connections sharing links, and the routing algorithm for different traffic classes.

The simulator manages connections as follows. An incoming request specifies the number of bytes to be sent and possibly the maximum rate the source can sustain for the request. Paths are selected by executing the routing algorithm and connections are set up; both operations can have a cost associated with them. Connections are torn down when the specified amount of data has been sent. The rates of all connections are dynamically adjusted when connections start and stop sending data.

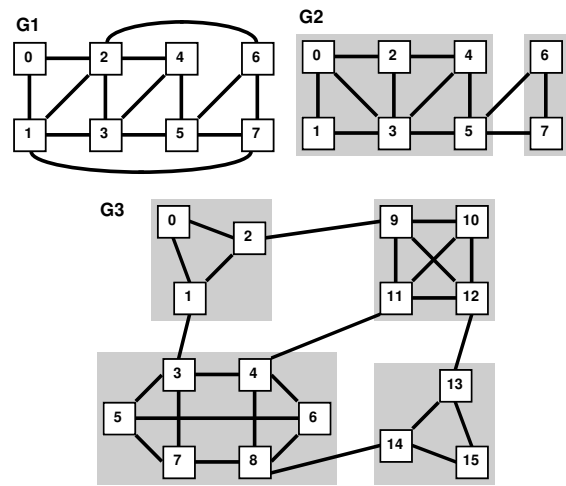


Figure 2: Topologies used by simulator

4.2 Simulation parameters

The main inputs to the simulator are the network topology, the traffic load, and the routing and connection management parameters.

We use three *topologies* $G1$, $G2$, and $G3$ (Figure 2) that have different degree of connectivity and size. For each of the topologies, we assume that eight host nodes are attached to each switch. The host-switch and switch-switch link bandwidth is 155 Mbit/second or 622 Mbit/second.

The nature of the *traffic* is controlled by a number of parameters. First, we distinguish two traffic types: low-latency and high-bandwidth traffic. The low latency traffic represents both the low-latency and rate-limited traffic from Section 2.2. The balance between the two traffic types is controlled by the parameter **HBFraction**, which represents the fraction of bytes of data sent that belong to high-bandwidth connections.

Second, the arrival rate of connection requests in each class follows a Poisson distribution, and the number of bytes in each request is uniformly distributed over [1KByte, **LLvsHB**] for low latency traffic and [**LLvsHB**, 1GByte] for high bandwidth traffic; most of the simulations use a threshold **LLvsHB** of 1 MByte. We believe this a good approximation of long-tail distribution of message sizes. For high-bandwidth requests, the source can make full use of bandwidth assigned by the network. For a low-latency connection, the source

specifies the maximum rate at which it can send data over the connection; this peak rate is in the range of 3 to 5 MByte/second.

Finally, the overall traffic load is modified by changing the average inter-arrival rate of connection requests and it is expressed as the average aggregate bandwidth of the traffic injected in the network at each switch by the hosts attached to it. We will consider both uniformly distributed and client-server loads. A uniform load means that each host has an equal chance of being a sender or a receiver, independent from the traffic type. In the client-server scenarios, the low-bandwidth load is still uniformly distributed, but most of the high-bandwidth load is between clients and servers. Servers are randomly selected from the pool of hosts at the start of the simulation. This represents the case of a distributed application (clients) making heavy use of a high-performance parallel file system (servers).

The *routing algorithms* used are widest-shortest path routing for low-bandwidth traffic and the algorithms described in Section 3 for high-bandwidth traffic. Initially we assume that routing algorithms have immediate access to the rate information, and we then look at the impact of using more realistic periodic updates that introduce a delay. Both the routing and the connection costs are included in the elapsed time of the connection since they are in the critical path of getting the data to the receiver. We use connection set up and tear down costs of 3 msec/hop and 1 msec/hop, respectively, while the routing cost for high-bandwidth connections is 10 milliseconds. There is no routing cost for low-latency connections.

4.3 Presentation of results

The main performance measure reported by the simulator is the average throughput achieved by high-bandwidth connections. Since the throughput is a ratio, we take the weighted harmonic mean to average the throughput [15], using the message length as the weight:

$$\overline{throughput} = \frac{\sum_{i \in N} b_i}{\sum_{i \in N} t_i} \qquad (4)$$

where b_i represents the number of bytes sent over connection i, and t_i its duration. Since the total number of bytes $\sum_{i \in N} b_i$ sent over the network is almost fixed for a long time interval, the $\overline{throughput}$ can also be viewed as a measure of elapsed time experienced by all high-bandwidth connections.

However, the average throughput is only a part of the picture. Another important parameter is how the throughputs are distributed. In general, having all throughputs fall in a small range is preferable over having a wide variance. This is especially true in a max-min fair share network that tries to balance the throughput of connections sharing links. For this reason, we will also discuss the actual throughput distribution for a few typical scenarios in more detail.

5 Simulation results for single-path routing

We compare the performance of the five routing algorithms discussed in 3 with different topologies and traffic load distributions. We also discuss how the routing algorithm affects the throughput distributions.

5.1 Average throughput as a function of traffic load

Figure 3 shows for all three topologies, the average throughput achieved by high-bandwidth connections as a function of the aggregate traffic arrival rate from all hosts connected to a switch. The traffic load is uniformly distributed with 90% of the bytes traveling over high-bandwidth connections. For all three topologies we can distinguish three phases corresponding to low, medium, and high traffic loads. We first focus on topology G1.

When the load is low, all algorithms give fairly similar performance, although "greedy" users who use the Shortest-widest, Shortest-dist$(P,2)$, or Shortest-dist$(P,1)$ path algorithm achieves slightly higher throughput. This result matches our intuition: when the network is lightly loaded, we expect all algorithms to perform well, but greedy algorithms are likely to have an edge.

As the network load increases, the average per-connection throughput decreases. The greedy shortest-widest path algorithm has the biggest drop in performance, while the Shortest-dist$(P,1/2)$ and widest-shortest path algorithms, which place more emphasis on finding a short path rather than a wide path, exhibit the slowest decrease in average throughput. The intuition is that with a higher network load, resources become more scarce, and algorithms that tend to pick longer paths (i.e. attach less value to a small hop count) perform more poorly. While a greedy algorithm might be able to increase the throughput of an individual connection by picking a long path with higher bandwidth, this might reduce the throughput of many other connections, and thus the average throughput. Moreover, paths with more hops have a higher chance of having their throughput reduced as a result of fair sharing with connections that are added later. The shortest-dist$(P,1)$ outperforms all the other algorithms.

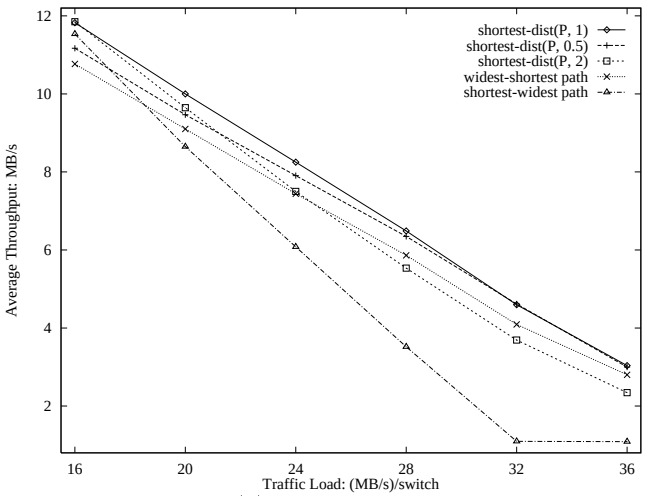
Further increases in network load reduce the difference in performance achieved by different algorithms. The reason is that under high load, all links are likely to be congested, so path selection becomes less sensitive to the obtainable rate and most algorithms tend to pick widest-shortest paths.

5.2 Impact of topology

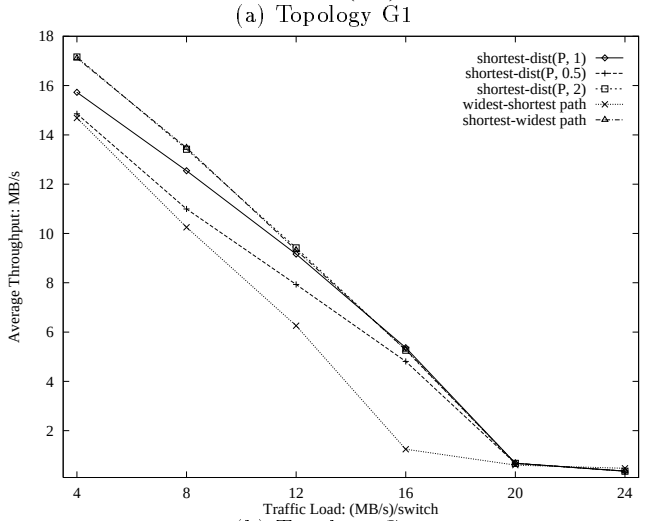
While the curves for the other topologies have a similar shape, there are some interesting differences. Topology $G2$ is less symmetric and has a lower degree of connectivity than topology $G1$. Figure 3(b) shows that as a result, greedy algorithms perform consistently better than algorithms that attach more weight to minimizing the number of hops. For example, the widest-shortest path, which performed well on topology $G1$, has very poor performance, and the shortest-widest and Shortest-dist$(P,2)$ path algorithms, which performed poorly on topology $G1$, give the best performance. This difference is a result of the unbalanced nature of topology $G2$: to make good use of the links connected to switch node 5 it is important to attach a lot of weight to the width of the path so that the bottleneck link can be avoided (link 3-5). The Shortest-dist$(P,1)$ path algorithm continues to perform well, e.g., it outperforms the widest-shortest path algorithm by as much as a factor of 4, while its throughput is only 9% or less lower than with the shortest-widest paths.

For topology $G3$ (Figure 3(c)) most algorithms have very similar performance for all loads. For example, shortest-dist$(P,1)$ can only perform 8% better than widest-shortest path. This is a result of the high degree of connectivity in $G3$, which results in a balanced load across links. The shortest-widest path algorithm is an exception: it performs poorly for the medium and high loads because it is too resource intensive.

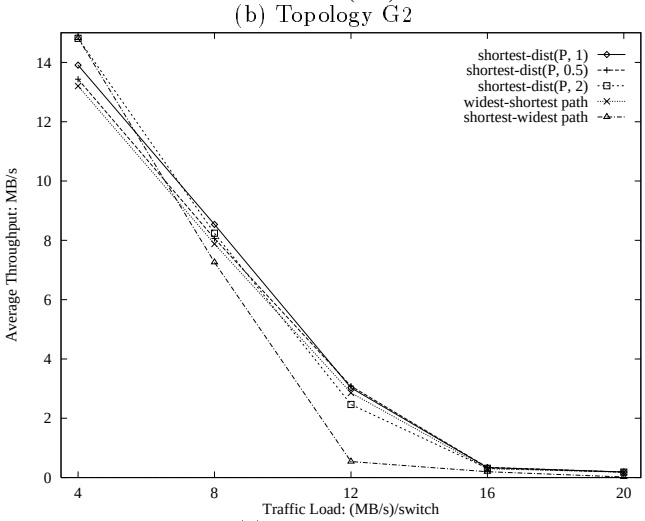
In summary, the Shortest-dist$(P,1)$ path algorithm consistently performs well across the different topologies and



(a) Topology G1



(b) Topology G2



(c) Topology G3

Figure 3: 90% bytes in high-bandwidth traffic

outperform the shortest-widest or widest-shortest path algorithms by as much as a factor of three. The reason is that it balances the "shortest path" and "widest path" metrics. Algorithms that favor one or the other metric tend to have a more uneven performance across the different topologies,

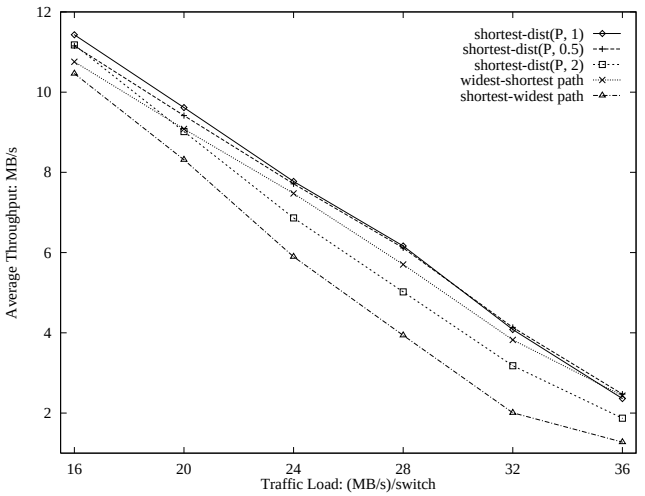


Figure 4: $G1$: 50% bytes in high-bandwidth traffic

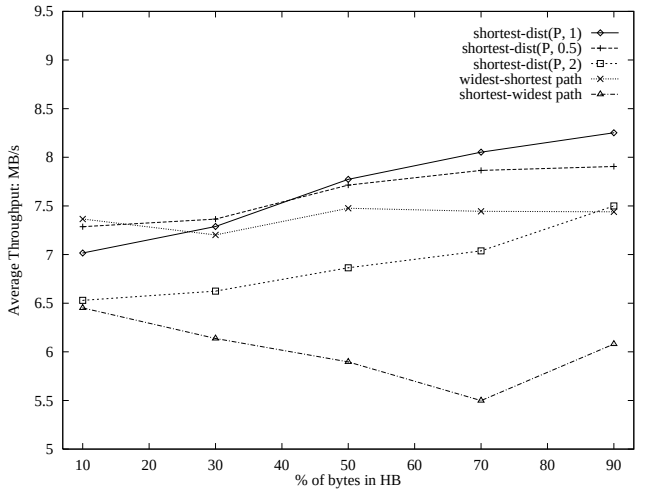


Figure 5: $G1$: arrival rate 24 MB/s per switch

especially for medium loads.

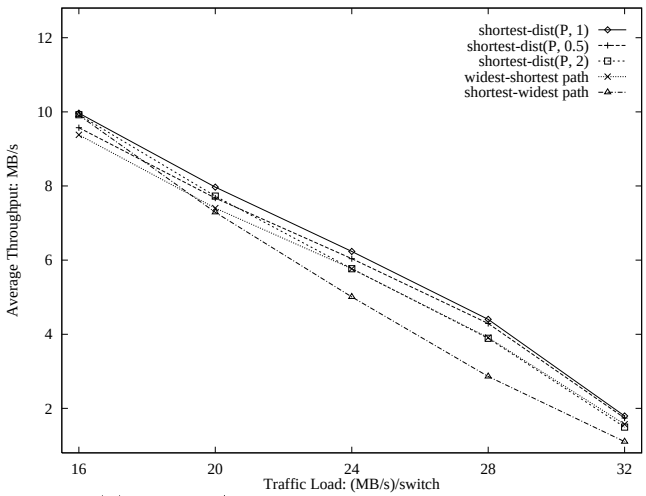
5.3 Impact of high-bandwidth traffic volume

We examine the effect of changing the distribution of traffic between high-bandwidth and low-latency connections. In Figure 4, the ratio of high-bandwidth traffic is reduced to 50%, compared to 90% in Figure 3(a). We observe that the results are similar, although the performance of the shortest-widest path is somewhat better. The Shortest-dist$(P, 1)$ path algorithm still outperforms the other algorithms.

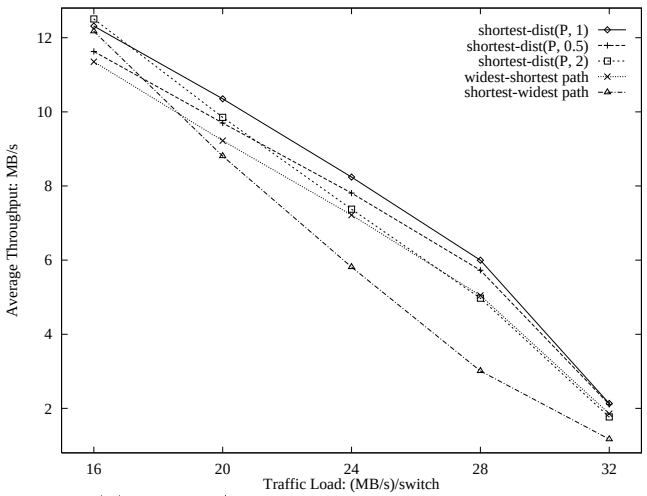
Figure 5 shows the average throughput as a function of the percentage of high-bandwidth traffic, for a fixed traffic load of 24 MB/s per switch. We see that as the contribution of high bandwidth traffic increases, the choice of routing algorithm used for high-bandwidth traffic has more impact, although even with only 10% high-bandwidth traffic, the best algorithm (widest shortest) still gives a 20% higher throughput than the worst algorithm (shortest widest).

5.4 Client-server configurations

The results so far used uniformly distributed traffic loads. We now split the 64 host nodes into 52 clients and 12 servers.

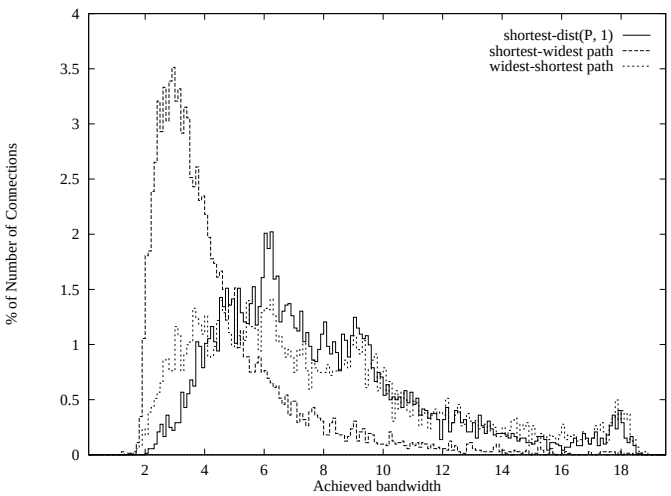


(a) 155 Mb/s server access link bandwidth

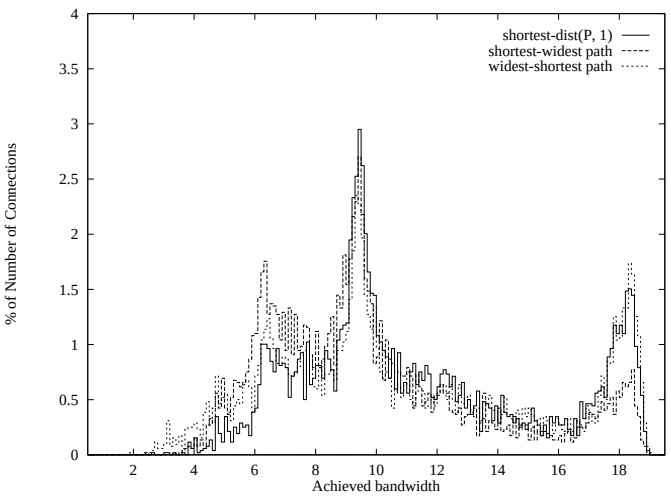


(b) 622 Mb/s server access link bandwidth

Figure 6: $G1$: 52 clients and 12 servers, 90% bytes in HB



(a) load of (28 MB/s)/switch



(b) load of (20 MB/s)/switch

Figure 7: $G1$: 90% bytes in HB

Most high-bandwidth traffic (90%) is between clients and servers, with the remaining 10% flowing between clients or between servers. Figure 6 shows results for server node access links of 155 Mb/s and 622 Mb/s; the client access links remain at 155 Mb/s. In both cases the shapes of the performance curves are similar to the uniform traffic load scenarios (Figure 3(a)), although the scale of the performance difference depends on the load distribution. For example, the shortest-dist($P, 1$) path algorithm outperforms the shortest-widest path algorithm by as much as 63% and widest-shortest path algorithm by as much as 14% (155 Mb/s); the differences are 100% and 20% for 622 Mb/s.

5.5 Variability of per-connection throughput

So far we have focused on the average throughput obtained by high-bandwidth connections. In this section we look at how the routing algorithm influences the throughput variability. Note that since the shape of the throughput distribution is often uneven and influenced by the topology, measures such as variance are not meaningful, so we exam the actual distribution.

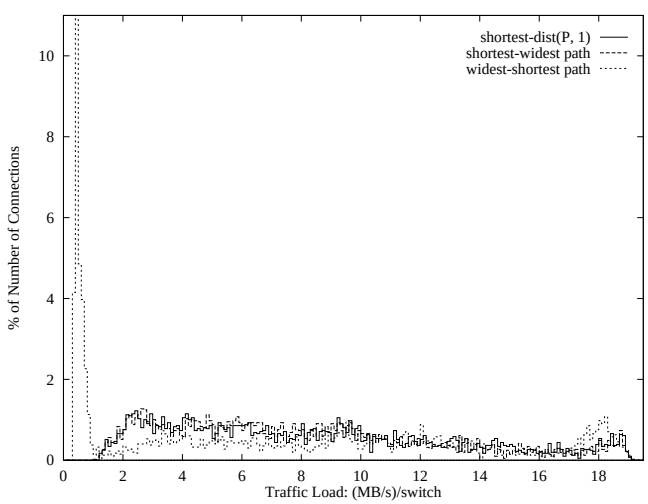


Figure 8: $G2$: 90% bytes in HB and (16 MB/s)/switch

In Figure 7, we present the throughput distribution of

high-bandwidth connections for the shortest-widest path, widest-shortest path, and Shortest-dist$(P,1)$ path algorithms. The results are for topology $G1$ with 90% high-bandwidth traffic and for two traffic loads: 28 MB/s and 20 MB/s per switch (compare with Figure 3(a)).

For the higher load, the throughput distribution for shortest-widest paths has has a peak around 3 MB/s and a long tail which corresponds to connections that obtain high throughput. With the the widest-shortest path and Shortest-dist$(P,1)$ path algorithms, the throughput is more evenly distributed between 3 to 9 MB/s, with a tail of higher throughput. With the shortest-widest path algorithm, few connections are able to get a high throughput because paths with more hops have a higher chance of having to share bandwidth with many connections. This can be seen from table 1, where we break down all connections according to the number of hops in their paths and show the average throughput, average initial rate, and distribution of connections with different hops.

hops	3	4	5	6	algorithms
throughput	4.4	3.3	2.9	2.7	
AveInitRate	4.2	3.3	3.1	3.6	shortest-widest
connections	30%	36%	23%	9%	
throughput	7.4	6.1	5.8	5.4	
AveInitRate	6.7	6.1	6.5	7	shortest-dist(P,1)
connections	46%	41%	12%	1%	

Table 1: Average throughput, average initial rate, and % of connections with different hops

When the network load is lower (20 MB/s per-switch in Figure 7(b)), the throughput distributions for the different algorithms are more similar. All three loads have approximately a bimodal distribution, which is a result of the network topology. Using the Widest-shortest and Shortest-dist$(P,1)$ path algorithms increases the chance of achieving very high throughput.

Figure 8 shows the throughput distribution for topology $G2$, with a traffic load of (16MB/s)/switch and 90% of high-bandwidth traffic (compare with Figure 3(middle)). It shows why the Widest-shortest path algorithm performs poorly: many widest-shortest paths use the link between switches three and five, resulting in a bottleneck and low throughput (0.5MB/s). The other two algorithms avoid the bottleneck and have more evenly distributed throughputs.

6 Simulation results for multi-path routing

We now move on to the evaluation of multi-path routing. Since our evaluation of single path routing algorithms shows that the Shortest-dist$(P,1)$ path algorithm has the best overall performance, we will only consider that algorithm at every priority level of multi-path routing. We will also limit our study to 2-path routing since our simulations show that the benefit of using a third and fourth path is limited (about an additional 5% increase in throughput).

Our main performance measure is the average throughput of high-bandwidth connections using multi-path routing, compared to that using single-path routing. Note that multi-path routing can improve throughput not only by adding a second path, but also by improving the throughput of the first path. The reason is that 2-path connections will often finish faster compared with single-path routing, thus freeing up bandwidth. To show this effect, we will also present the average throughput for 1-path and 2-paths connections separately.

6.1 Average throughput as a function of traffic load

Figure 9 shows the average throughput as a function of the traffic load for two different percentages of high-bandwidth traffic, 50% and 90%. The results are for topology $G1$, but similar results were observed for the other topologies. We observe that 2-path routing increases the average throughput compared with single-path routing, not only for connections that use two paths but also for connections that use a single path. We observe that, as the traffic load becomes higher, the increase in throughput gets smaller, although the relative increase in throughput with multi-path routing compared with single-path routing remains relatively constant.

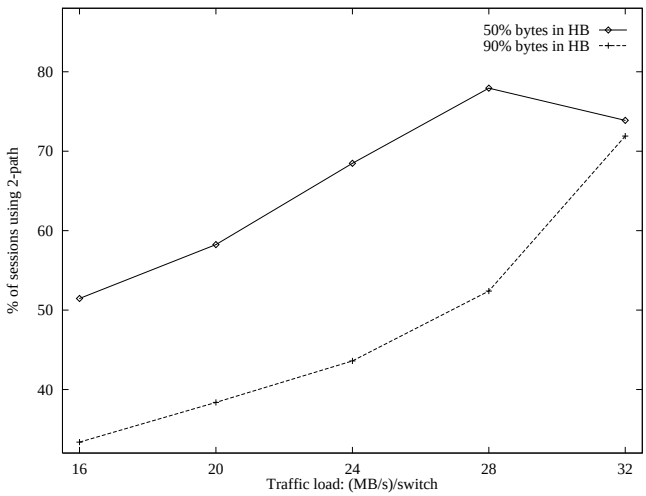


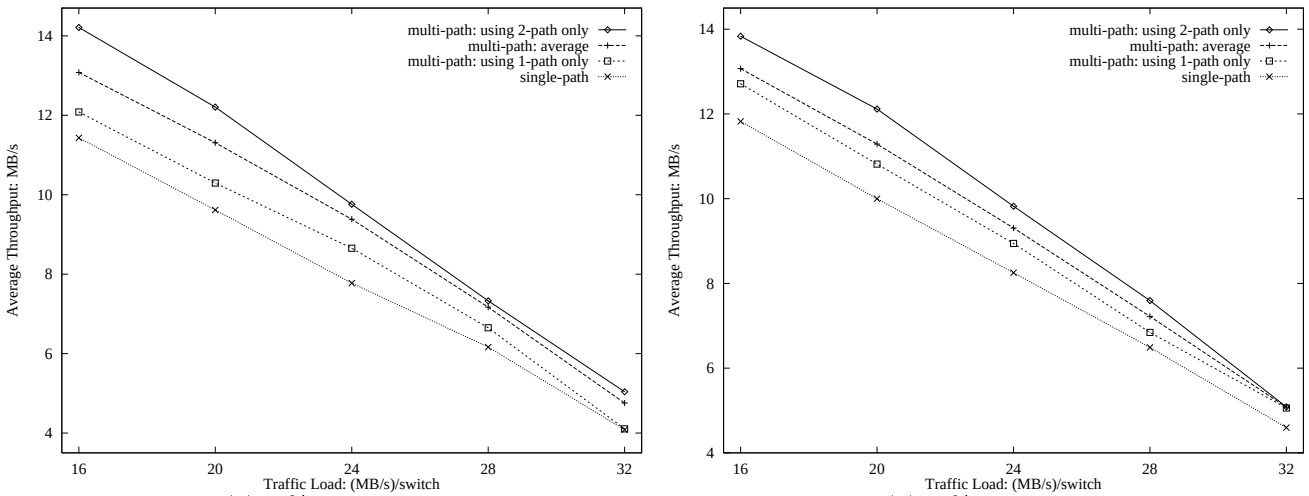
Figure 10: $G1$: percentage of sessions using two paths

Figure 10 shows that the percentage of connections that use two paths increases with the traffic load. This is a result of the fact that, when the traffic load becomes higher, the shortest-dist$(P,1)$ path algorithm tends to pick the shortest path more often, which leads to a relatively higher number of links with unused bandwidth, and these links can accommodate more secondary paths due to the increased number of arrivals.

6.2 Impact of high-bandwidth traffic volume

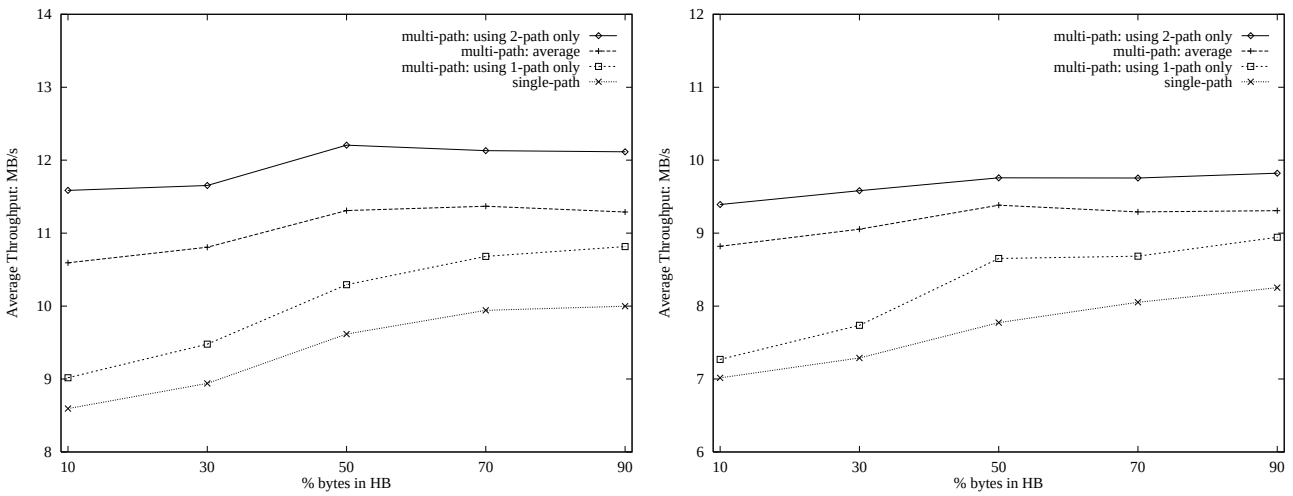
Figure 11 shows the average throughput using single-path and 2-path routing as a function of the percentage of high-bandwidth traffic for a traffic load of 20 MB/s and 24 MB/s per switch. Connections that use two paths achieve an average increase in throughput of 20% to 35%, while connections that use a single path have a increase of 2% to 8%. The overall improvement ranges from 13% to 26%. We also observe that the benefit of multi-path routing decreases as the contribution of high-bandwidth traffic increases. The reason is that more high-bandwidth traffic results in more competition among secondary paths.

Figure 12 shows the percentage of sessions that actually use two paths for the two scenarios in Figure 11. The percentage of sessions that use two paths decreases as the high-bandwidth traffic volume increases (although the number of 2-path connections goes up). The reason is that high-bandwidth connections can use any available network bandwidth, i.e. they can by themselves saturate links, making them unavailable for secondary paths. As a result, more



(a) 50% HB traffic (b) 90% HB traffic

Figure 9: $G1$: average throughput as a function of traffic load for multipath routing



(a) 20 MB/s per switch (b) 24 MB/s per switch

Figure 11: $G1$: average bandwidth as a function of the percentage of high-bandwidth traffic

high-bandwidth traffic means fewer links available for secondary paths and a lower percentage of 2-path connections.

Our results suggest that multi-path routing is an effective technique to make use of unused network resources in a max-min fair share network. While the performance improvement for multi-path routing is relatively constant across different traffic loads (previous section), it is sensitive to the volume of high-bandwidth traffic. When the percentage of high-bandwidth traffic increases the performance improvement from multi-path routing goes down, both because it is harder to find secondary paths and because less bandwidth is available once they are established.

6.3 Variance of increased throughput

Figure 13 shows the distribution of the throughput increase over single-path routing for sessions that use two paths; the graph includes results 30% and 70% high-bandwidth traffic (compare to Figure 11). The two distributions are fairly symmetric, with an average throughput increase around 3 MB/s. A few sessions increased their throughput by as much as 6 to 10 MB/s. A few sessions suffer a throughput reduction. The reason is that the early completion of some ses-

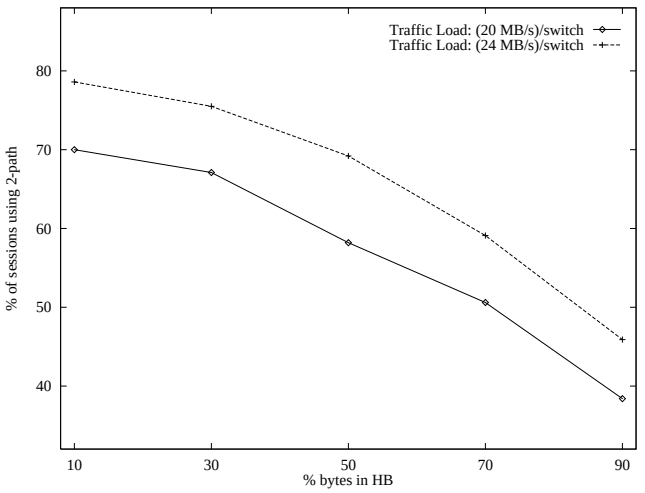


Figure 12: $G1$: arrival rate 24 MB/s per switch

sions changes the routes of later sessions, and in some cases that results in routes with a slightly lower average rate.

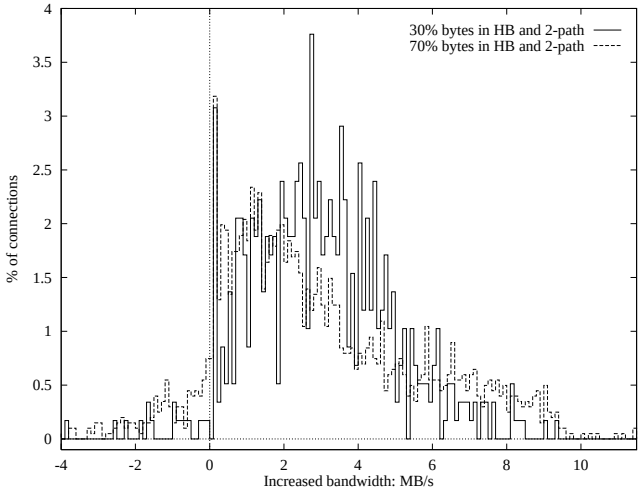


Figure 13: $G1$: throughput increase for two-path connections with an arrival rate 20 MB/s per switch

Figure 14 shows the distribution of the throughput increase over single-path routing for sessions that could not find a second path; the peak (off scale) corresponds to 54% of the connections observing an increase of about 0.1 MB/s. On average, single-path connections benefit slightly from multipath routing. We also observed that the distribution is more spread out when the ratio of high-bandwidth traffic is higher. This indicates that there is more interference among high-bandwidth connections.

7 Sensitivity analysis

In the previous discussions, we assumed that accurate routing information is available whenever making a routing decision, and used fixed PCRs (3-5 MB/s depending on message size) for low-latency traffic, and a fixed routing cost of 10 ms for routing algorithms other than the widest-shortest path. In this section, we show the performance impact of changing these parameters.

7.1 Impact of routing information update interval

In Figure 15, we show a scenario with the same traffic condition and topology as Figure 3 (a), but with a 100ms routing information update interval, i.e., routing information is usually somewhat dated. The two figures are very similar, although a lightly worse performance for the one 100ms routing update interval can be observed, especially for the shortest-widest path algorithm. This suggests that greedy algorithms might be more sensitive to outdated information.

One potential problem with load-sensitive routing is that it might lead to oscillation. This is specifically a problem when the frequency of status updates is low compared with the rate of change in the network. The reason is that out-of-date load information can result in traffic being directed to some part of the network, even after it has become already heavily loaded, while other, previously heavily loaded links, are relatively lightly loaded; this trend is then reversed after the next update. We do not expect this to be a problem for high-bandwidth routing. The reason is that the changes in high-bandwidth connections are, almost by definition, relatively infrequent, since there are relatively few

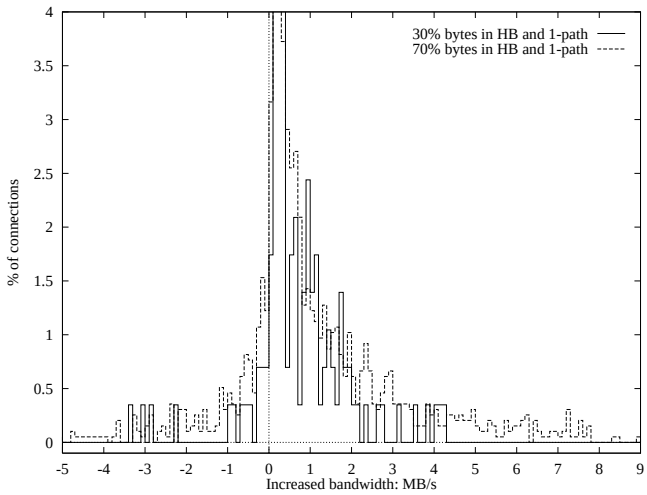


Figure 14: $G1$: throughput increase for single-path connections with an arrival rate 20 MB/s per switch

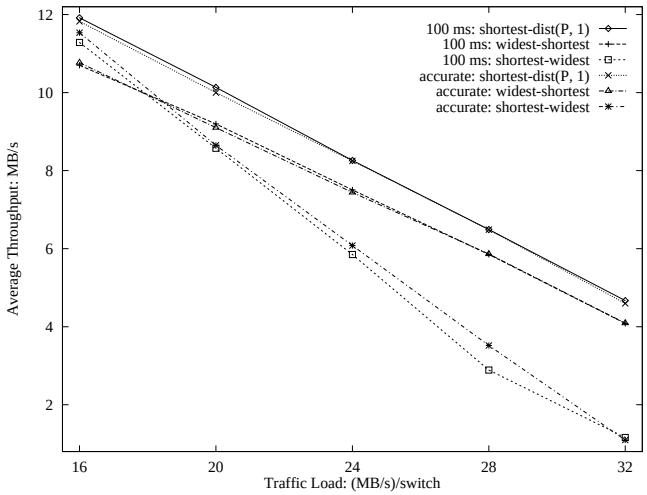


Figure 15: G1: 90% bytes in HB traffic and different routing update interval

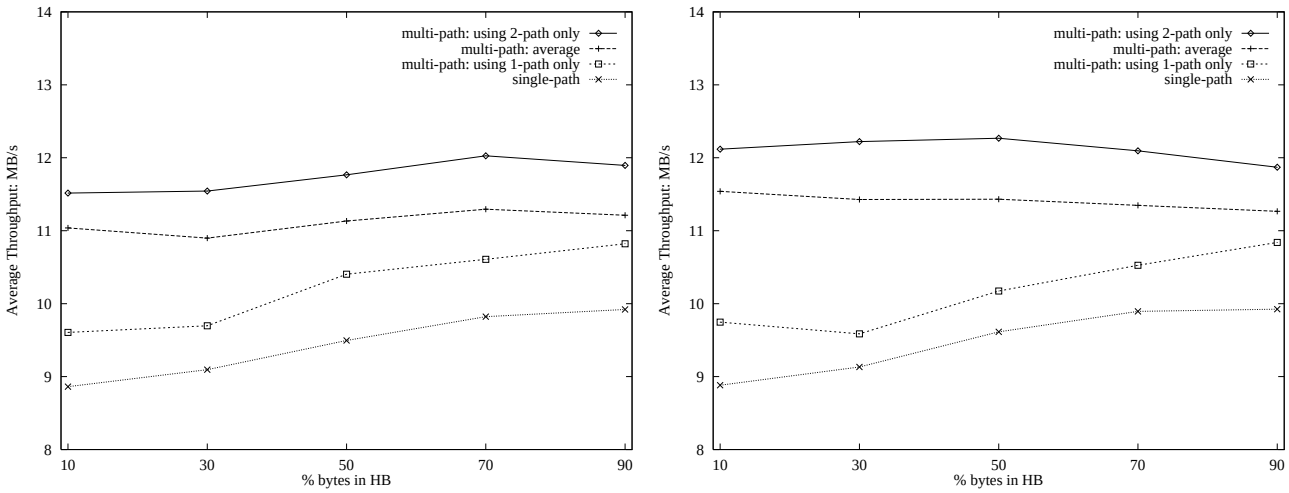
high-bandwidth connections and they are long-lived. As a result, the connectivity and bandwidth information does not get stale fast, and even infrequent periodic routing updates are likely to be sufficient to avoid oscillation.

7.2 Impact of PCR of low-latency traffic

In Figure 16, we show the performance difference for multi-path routing by increasing (a) and decreasing (b) PCR rate for low-latency traffic by 1 MB/s. The topology is G1 and the traffic load is 24 MB/s per switch. We observe that the performance of single-path routing seems fairly insensitive to the PCR. For multi-path routing, while the overall performance is very close to what we showed earlier (Figure 11(a)), the performance improvement is slightly higher when the PCR is lower. The reason is that a lower PCR leaves higher unused bandwidth to lower priority paths.

7.3 Impact of routing cost

In Table 2 we list average throughputs for two different routing costs (1ms instead of 10ms); the results are for topology



(a) PCR 4-6 MB/s PCR 2-4 MB/s

Figure 16: $G1$: arrival rate 24 MB/s per switch

$G1$, a ratio of high-bandwidth traffic of 90%, and a traffic load of 24 MB/s per switch. We see that the change in routing cost has little impact on the results.

	1 ms	10 ms
shortest-widest	5.86	6.08
widest-shortest	7.45	7.44
shortest-dist(P, 1)	8.29	8.25
shortest-dist(P, 2)	7.45	7.50
shortest-dist(P, 0.5)	7.91	7.91

Table 2: Average throughput in MB/s for two routing costs

7.4 Impact of LLvsBB

In Table 3, we list average throughputs for two different cutoff **LLvsBB**'s between high-bandwidth and low-latency traffic. The results are for topology $G1$, a **HBFraction** of 90%, and a traffic load of 28 MB/s per switch. The performance becomes slightly worse when **LLvsBB** increases from 1 to 10 MByte, due to the decreased number of connections and consequently increased traffic concentration. The impact on the results is little, although it affects more on shortest-widest paths than on shortest-dist(P,1) paths.

	1 MB	10 MB
shortest-widest	3.518	3.177
widest-shortest	5.863	5.588
shortest-dist(P, 1)	6.488	6.417
shortest-dist(P, 2)	5.535	5.495
shortest-dist(P, 0.5)	6.346	6.123

Table 3: Average throughput in MB/s for two **LLvsHB**'s

8 Related work

To the best of our knowledge, this paper is the first that studies routing algorithms in networks with max-min fair sharing. In this section, we review related work in the areas of service definition and routing.

It has been long recognized that data communication applications can be divided into several classes, including bulk data transfer and interactive applications. However, until recently, networks did not distinguish between these classes. With the advent of integrated service networks, several proposals [8, 21, 17] have been made to divide the traditional best-effort service into multiple service classes. An alternative to defining service classes is to implement queueing policies that optimize the performance of interactive application without sacrificing bulk data transfer applications; this is for example done in the DataKit network [12]. These approaches rely on traffic management support to optimize the performance of different classes of applications. In contrast, we assume that the traffic management algorithms treats data transfers in the same way, and we pursue the use of routing to optimize performance.

Packet-switched networks have traditionally used shortest-path routing. While different measured "link-cost" measures can be used (see [20, 18]), earlier networks typically selected minimal-hop paths. The problem with using measured link costs is that it does not always accurately account for how resources are shared among connections, so they can be inaccurate and even misleading. The rate information we use is an accurate measure of available bandwidth since we model the sharing algorithm that is used in the max-min fair share network.

Routing in circuit-switched networks has focused on finding paths with certain quality-of-service (QoS) guarantees while minimizing the blocking rate of future requests. Trunk-reservation [1], adaptive routing [13], shortest-widest path [26], and min-max routing have been well-studied and are very relevant to today's QoS routing in data networks. However, these algorithms are based on a residual bandwidth model, which is representative for reservation-based networks but not for max-min share networks.

Multipath routing algorithms have been used to optimize network performance [19, 25, 2, 5, 22, 24]. Multiple paths are selected in advance. When data traffic arrives, a path with the lowest traffic load is used. None of these studies addresses the issue of fairness. In contrast, we studied the use of multiple paths simultaneously to maximize throughput in a max-min fair share network.

9 Conclusions

In this paper, we studied routing support for high-bandwidth traffic. in max-min fair sharing networks. A fundamental feature of our routing algorithms is that they make use of

rate information provided by the fair share congestion control mechanism. By giving the routing algorithm access to rate information, we couple the coarse grain (routing) and fine grain (congestion control) resource allocation mechanisms, allowing us to achieve efficient and fair allocation of resources.

Our evaluation of single-path routing algorithms for high-bandwidth traffic shows that the **Shortest**-dist$(P, 1)$ path algorithm performs best in most of the situations we simulated. While the **Shortest-widest** path algorithm can give slightly better performance when the network load is very light, it can have very poor performance for medium and high traffic loads, because it tends to pick paths that are resource-intensive. The **Shortest**-dist$(P, 1)$ path algorithm is able to route around bottlenecks, thus avoiding the clusters of connections with very low throughput that are sometimes the result of using the **widest-shortest** path algorithm. Overall, the **Shortest**-dist$(P, 1)$ path algorithm balances the weight given to the "shortest" and "widest" metrics in an appropriate way.

Finally, we introduce a prioritized multi-level max-min fairness model, in which multiple paths are assigned different priority. This approach prevents a multi-path connection from grabbing an unlimited amount of bandwidth by using a large number of paths, i.e. additional paths use only unused bandwidth and do not affect the bandwidth available to primary paths. Our simulations show that 2-path routing increases the average bandwidth compared with single-path routing by 25% overall and 35% for those connections using two paths.

Acknowledgements We would like to thank Jon Bennett, Allan Fisher, Garth Gibson, Kam Lee, Bruce Maggs, K.K. Ramakrishnan, and Lixia Zhang for helpful discussions.

References

[1] J.M. Akinpelu. The Overload Performance of Engineered Networks with Nonhierarchical and Hierachical Routing. *Bell System Technical Journal*, pages 1261–1281, September 1984.

[2] S. Bahk and M. Elzarki. Dynamic Multipath Routing and How it Compares with Other Dynamic Routing Algorithms for High Speed Wide Area Networks. *ACM SIGCOMM 92*, September 1992.

[3] J. Behrens and J.J. Garcia-Luna-Aceves. Distributed, Scalable Routing Based on Link-State Vectors. *ACM SIGCOMM*, September 1994.

[4] F. Bonomi and K. Fendick. The Rate-Based Flow Control Framework for the Available Bit Rate ATM Service. *IEEE Network*, 9(2):25–39, March/April 1995.

[5] L. Breslau, D. Estrin, and L. Zhang. A Simulation Study of Adaptive Source Routing in Integrated Service Networks. *USC CSD Technical Report*, Sep., 1993.

[6] A. Charny, D.D. Clark, and R. Jain. Congestion Control With Explicit Rate Indication. In *Proc. ICC'95*, June 1995.

[7] A. Charny, K.K. Ramakrishnan, and A. Lauck. Scalability Issues for Distributed Explicit Rate Allocation in ATM. In *Proc. IEEE INFOCOM'96*, 1996.

[8] David D. Clark. Adding Service Discrimination to the Internet. Preprint, 1995.

[9] 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.

[10] PNNI SWG Doug Dykeman (ed.). PNNI Draft Specification. ATM Forum 94-0471R10, October 1995.

[11] E.W. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, pages 1:269–271, 1959.

[12] A. Fraser. Towards a Universal Data Transport System. *IEEE JSAC*, pages 803–816, Nov 1983.

[13] 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.

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

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

[16] K. Lang and S. Rao. Finding Near-Optimal Cuts: An Empirical Evaluation. In *Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 212–221, 1993, Austin, Texas.

[17] J. Liebeherr, I.F. Akyildiz, and A. Tai. A Multi-level Explicit Rate Control Scheme for ABR Traffic with Heterogeneous Service Requirements. *Submitted for Publication*, July 1995.

[18] J.M. McQuillan, I. Richer, and E. Rosen. The New Routing Algorithm for the ARPANET. *IEEE Transactions on Communications*, COM-28(5):711–719, May 1980.

[19] D.J. Nelson, K. Sayood, and H. Chang. An Extended Least-hop Distributed Routing Algorithm. *IEEE Transactions on Communications*, COM-38(4):520–528, April 1990.

[20] M. Schwartz and T.E. Stern. Routing Techniques Used in Computer Communication Networks. *IEEE Transactions on Communications*, COM-28(4), April 1980.

[21] S. Shenker, D.D. Clark, and L. Zhang. A Scheduling Service Model and a Scheduling Architecture for an Integrated Service Packet Network. *preprint*, 1993.

[22] J. Sole-Pareta, D. Sarkar, J. Liebeherr, and I.F. Akyildiz. Adaptive Multipath Routing of Connectionless Traffic in an ATM Network. In *Proc. IEEE ICC'95*, May 1995.

[23] M. Steenstrup. Inter-Domain Policy Routing Protocol Specification: Version 1. Technical Report Internet Draft, May 1992.

[24] H. Suzuki and F. A. Tobagi. Fast Bandwidth Reservation Scheme with Multi-link and Multi-path Routing in ATM Networks. In *Proc. IEEE INFOCOM'92*, 1992.

[25] Z. Wang and J. Crowcroft. Shortest Path First with Emergency Exits. *ACM SIGCOMM 90*, September 1991.

[26] Z. Wang and J. Crowcroft. QoS Routing for Supporting Resource Reservation. *IEEE JSAC*, to appear 1996.