A New Class of Online Minimum-Interference Routing Algorithms

Ilias Iliadis and Daniel Bauer

IBM Research, Zurich Research Laboratory, 8803 Rüschlikon, Switzerland
{ili,dnb}@zurich.ibm.com

Abstract. On-line algorithms are essential for service providers to quickly set up bandwidth-guaranteed paths in their backbone or transport networks. A minimum-interference routing algorithm uses the information regarding the ingress–egress node pairs for selecting a path in the case of on-line connection requests. According to the notion of minimum interference, the path selected should have a minimum interference with paths considered to be critical for satisfying future requests. Here we introduce a new class of minimum-interference routing algorithms, called "simple minimum-interference routing algorithms" (SMIRA), that employ an efficient procedure. These algorithms use static network information comprising the topology and the information about ingress–egress node pairs, as well as the link residual bandwidth. Two typical algorithms belonging to this class are introduced, and their performance is evaluated by means of simulation. The numerical results obtained illustrate their efficiency, expressed in terms of throughput, and fairness.

1 Introduction

This paper deals with the issue of dynamic bandwidth provisioning in a network. This problem arises in several instances, such as in the context of dynamic label-switched path (LSP) setup in multiprotocol label switching (MPLS) [1] networks and in the context of routing virtual circuit requests over an ATM backbone network [2]. In particular, this paper considers the issue of establishing bandwidth-guaranteed connections in a network, in which connection-setup requests arrive one by one and future demands are unknown. This is referred to as an *on-line* algorithm, in contrast to an *off-line* algorithm that assumes *a priori* knowledge of the entire request sequence, including future requests. On-line algorithms are essential owing to the need of service providers to quickly set up bandwidth-guaranteed paths in their backbone or transport networks.

The primary routing problem consists of determining a path through the backbone network that a connection should follow. Clearly, the available bandwidth of all links on the path should be greater or equal to the requested bandwidth. If there is insufficient capacity, some of the connections cannot be established, and therefore are rejected. A significant body of work exists for the on-line path selection problem. Several path selection algorithms proposed in

E. Gregori et al. (Eds.): NETWORKING 2002, LNCS 2345, pp. 959–971, 2002.

the literature aim at limiting the resource consumption so that network utilization is increased. The most important of these algorithms can be found in [3, 4]. The basic algorithms considered here along with a short description of their functionality are listed below. **Shortest-path routing** (SP) algorithms select a path with the least amount of aggregated cost. **Minimum-hop routing** algorithms select a path with the least number of links as that path uses the smallest amount of resources. When all links have the same cost, it is a special case of an SP algorithm. **Widest-shortest-path routing** (WSP) algorithms select a shortest path with the largest available (residual) bandwidth. **Shortest-widest-path routing** (SWP) algorithms select a widest path with the least amount of aggregated cost. A widest path is a path with maximum bottleneck bandwidth or, equivalently, a path with the largest In addition to these algorithms, a more sophisticated algorithm, called **minimum interference routing algorithm (MIRA)**, that uses the information regarding the ingress–egress pairs was recently developed [5]. Despite the fact that this algorithm uses this information pertaining to the past, present, and future requests, it is considered to be an on-line algorithm because the future connection requests are unknown. However, the effect of future requests is indirectly incorporated through the notion of the *minimum-interference routing*.

The second problem related to the primary routing is that of admission control. Routing algorithms can be categorized into two classes according to the control of admitting connections [6]. *Greedy* algorithms always establish a connection as long as sufficient capacity is available. *Trunk-reservation* algorithms reject a connection if assigning any of the existing paths could result in inefficient use of the remaining capacity regarding future connection requests [7].

Section 2 reviews the concept of minimum-interference routing and briefly describes the MIRA algorithm, the first such algorithm presented in [5]. The main contributions of this paper are presented in Sections 3 and 4. In Section 3, we present a new class of minimum-interference routing algorithms, called "simple minimum-interference routing algorithms" (SMIRA). This class of algorithms is not based on the principle of calculating maximum flows, but rather uses a more efficient (in terms of computational complexity) approach, hence the term "simple". In Section 4, we examine the efficiency of the algorithms proposed by means of simulation, and compare them with the SP, SWP, WSP, and MIRA algorithms. The efficiency assessment is based on performance metrics including the throughput, expressed as the total bandwidth of the routed (accepted) connections, the blocking-free range, and the fairness achieved among different ingress–egress node pairs. In particular, we demonstrate that the effectiveness of a given algorithm strongly depends on the performance criterion chosen. Finally, we draw conclusions in Section 5.

2 Minimum-Interference Routing

In this section, the notion of minimum-interference routing and the first MIRA algorithm are reviewed. Although for on-line algorithms future connection re-

quests are unknown, the effect of future requests can be indirectly incorporated through the notion of *minimum-interference routing*. A new connection should follow a path that does not "interfere too much" with a path that may be critical to satisfy a future request. Note that this notion can only be used in conjunction with knowledge of all ingress–egress pairs. An explicit path between a given ingress–egress pair can in principle be calculated according to a defined interference criterion. For example, the criterion could be the maximization of the smallest maxflow value of all remaining ingress–egress pairs, the maximization of the weighted sum of the remaining maxflows (referred to as WSUM-MAX) [5], or the maximization of the maximum throughput of the corresponding multicommodity flow problem [7]. These problems are quite complex, therefore alternative, simplified approaches are highly desirable. One possibility is to turn the original problem into an equivalent shortest-path routing problem with appropriately selected link-cost metrics. In [5], this transformation is done as follows. The amount of interference on a particular ingress–egress pair, say (s, d), due to the routing of a connection between some other ingress–egress pair is defined as the decrease of the *maximum flow* value between (s, d). Then, the notion of *critical links* is introduced. These are links with the property that whenever a connection is routed over them, the maxflow values of one or more ingress–egress pairs decrease. For this definition, it turns out that the set of critical links corresponding to the (s, d) pair coincides with the links of all *minimum cuts* for this pair. Links are subsequently assigned weights that are an increasing function of their "criticality": the weight of a link is chosen to be the rate of change in the optimum solution of the original WSUM-MAX problem with respect to changing the residual capacity of the link. This choice results in critical links being assigned the highest weights. Finally, the actual explicit path is calculated using an SP algorithm.

3 Simple Minimum-Interference Routing Algorithms

In this section, we introduce the class of the simple minimum-interference routing algorithms (SMIRA). The term "simple" reflects the fact that they do not define the critical links according to maximum-flow calculations, as done by MIRA, but rather employ a simpler approach, which, as will be seen below, has a lower computational complexity than the MIRA maximum-flow approach.

To devise an alternative notion for the critical links we resort to the fundamental objective of minimum-interference routing, namely that a new connection must follow a path that interferes as little as possible with a path that may be critical to satisfy a future request. This requires that paths associated with future requests be taken into account and also that links associated with such critical paths be identified and weighted accordingly. One possible way to achieve this is the following. Let P be the set of ingress–egress node pairs, and suppose that the connection request is between nodes a and b. For every of the remaining ingress–egress pairs (s, d) we identify a set of critical paths, and each link of such a path is weighted accordingly. When this process has been completed, links having

minimum weight are associated with paths that do not interfere with future requests. Routing the current connection request on a shortest path with respect to these weights results in a residual network in which the interference of the remaining ingress–egress pairs is kept to a minimum.

3.1 Critical Paths

There are several ways to identify a set of critical paths corresponding to the ingress–egress pair (s, d). Here we introduce a procedure for obtaining the set of critical paths called *K-widest-shortest-path under bottleneck elimination*. This procedure identifies a set of critical paths by making use of a WSP algorithm. The paths are enumerated in descending order of their significance. The algorithm starts with selecting the widest-shortest path between pair (s, d). Let $L_{sd}^{(1)}$ denote the set of links constituting this widest-shortest path, and let $btl_{sd}^{(1)}$ be the corresponding (bottleneck) bandwidth of this path. Let also $Btl_{sd}^{(1)}$ denote the subset of link(s) of this path whose residual bandwidth is equal to the bottleneck value $btl_{sd}^{(1)}$. The next (second) member is found by computing the shortest-widest path when the links of the set $Btl_{sd}^{(1)}$ are removed from the network. This procedure is repeated until either K paths are found or no more paths are available, whichever occurs first. It is realized by using the Dijkstra algorithm [8] in each iteration, therefore its complexity is of order $O(K(n \log n + m))$, n and m being the number of nodes and links, respectively.

An alternative procedure can be derived by using an SWP algorithm – or any other path-computation method – to enumerate the critical paths. This procedure, called *K-shortest-widest-path under bottleneck elimination*, is realized by using the Dijkstra algorithm twice in each iteration [3], therefore its complexity is also of order $O(K(n \log n + m))$.

For networks of practical relevance, it turns out that the value of K is typically a small number. Therefore, the complexity of the above procedures is of order $O(n \log n + m)$. On the other hand the complexity of the procedure for determining the set of critical links in the case of the MIRA algorithm is of order $O(n^2\sqrt{m} + m^2)$ [5]. This complexity results from the two phases used by MIRA. The first consists of a maximum flow calculation with a computational complexity of $O(n^2\sqrt{m})$. The second consists of the process of enumerating all links belonging to minimum cuts with a complexity of $O(m^2)$. Thus, the total complexity is of order $O(n^2\sqrt{m}+m^2)$. In particular, in the case of sparse topologies, MIRA's complexity is of order $O(n^{2+\frac{1}{2}})$ as m is of order $O(n)$, whereas that of our algorithm is of order $O(n \log n)$. In the case of dense topologies, MIRA's complexity is of order $O(n^4)$ as m is of order $O(n^2)$, whereas that of our algorithm is of order $O(n^2)$. Therefore, our proposed procedures have a shorter expected execution time, justifying the use of the term "simple".

3.2 Link-Weight Assignment

Each link is initially assigned a static cost. For critical links this cost is scaled by a factor that includes two weight components. The first one is associated with the path(s) to which this link belongs. Naturally, higher weights are assigned to paths with higher significance. The second reflects the importance of the links constituting each of the critical paths.

We now turn our attention to the first weight component. Let $L_{sd}^{(i)}$ denote the set of the links constituting the i-th path associated with the ingress–egress pair (s, d), and $w_{sd}^{(i,l)}$ denote the corresponding weight contributed to link l of this set. Let also $btl_{sd}^{(i)}$ be the corresponding bandwidth of this path, and $Btl_{sd}^{(i)}$ be the subset of the corresponding bottleneck link(s). The paths are enumerated in descending order of their significance. In accordance, the links of the $L_{sd}^{(i)}$ path are weighted with a factor $v_{sd}^{(i)}$, which is a decreasing function in i such that the weight for link l should be proportional to this factor, i.e. $w_{sd}^{(i,l)} \sim v_{sd}^{(i)}$. Our rationale is the following: if all candidate paths for the new connection contain links that have already been marked by this process, i.e. the interference cannot be avoided, then the links associated with the most critical paths corresponding to the other ingress–egress pairs should be avoided by assigning them the highest weights. In this way the interference is relegated to secondary paths. There are infinitely many discounting value functions one could choose. Here we consider the following two functions: $v_{sd}^{(i)} = 1$ and $v_{sd}^{(i)} = (K - i + 1)/K$.

The next consideration is the weight assignment for the links constituting path $L_{sd}^{(i)}$. Let $g_{sd}^{(i,l)}$ denote the corresponding weight for link l. Intuition dictates that bottleneck links should be assigned a higher value than other links. Here again, there are infinitely many discounting value functions one could choose. In this paper, we consider two functions defined as follows. The inversely proportional function $g_{sd}^{(i,l)} = btl_{sd}^{(i)}/r(l)$ and the step function $g_{sd}^{(i,l)} = \lfloor btl_{sd}^{(i)}/r(l) \rfloor$, where $r(l)$ denotes the residual bandwidth of link $l \in L_{sd}^{(i)}$. Note that $btl_{sd}^{(i)}/r(l)$ is a decreasing function in $r(l)$ with $btl_{sd}^{(i)}/r(l) \leq 1$. Consequently, $\lfloor btl_{sd}^{(i)}/r(l) \rfloor = 1$, if and only if $r(l) = btl_{sd}^{(i)}$, otherwise $\lfloor btl_{sd}^{(i)}/r(l) \rfloor = 0$. Thus, in the case of the step function, it holds that $g_{sd}^{(i,l)} = \begin{cases} 1 & l \text{ is a bottleneck link of the path } L_{sd}^{(i)} , \\ 0 & \text{otherwise} . \end{cases}$

We proceed by choosing the weight contributed to link l to be proportional to the factors defined above, i.e. $w_{sd}^{(i,l)} \sim v_{sd}^{(i)}$ and $w_{sd}^{(i,l)} \sim g_{sd}^{(i,l)}$. In particular, we choose $w_{sd}^{(i,l)} = w\, v_{sd}^{(i)}\, g_{sd}^{(i,l)}$, $(l \in L_{sd}^{(i)})$, where w is a scaling factor.

By taking into account all contributions, the weight of each link is then calculated by $w(l) = c(l) \left\{ 1 + \sum_{(s,d)\in P\setminus(a,b)} a_{sd} \sum_{i=1}^{K} \sum_{l\in L_{sd}^{(i)}} w_{sd}^{(i,l)} \right\}$, or

$$w(l) = c(l) \left\{ 1 + w \sum_{(s,d)\in P\setminus(a,b)} a_{sd} \sum_{i=1}^{K} v_{sd}^{(i)} \sum_{l\in L_{sd}^{(i)}} g_{sd}^{(i,l)} \right\}, \tag{1}$$

where $c(l)$ is a static cost that could, for example, depend on the capacity of the link, and a_{sd} is the weight of the ingress–egress pair (s, d). Note that for noncritical links it holds that $w(l) = c(l)$.

3.3 The SMIRA Algorithm

In summary the SMIRA algorithm is as follows:
INPUT: A graph $G(N, L)$, the residual bandwidth $r(l)$ for each link, and a set P of ingress–egress node pairs. An ingress node a and an egress node b between which a flow of D units have to be routed.
OUTPUT: A route between a and b with a bandwidth of D units, if it exists.
ALGORITHM:
1. Compute the K-critical paths $\forall(s, d) \in P\backslash(a, b)$. Let $L_{sd}^{(i)}$ be the set of the links constituting the i-th path.
2. Assign weight on each link according to Eq. (1).
3. Eliminate all links that have residual bandwidth smaller than D and form a reduced network.
4. Use Dijkstra's [8] algorithm to compute the shortest path in the reduced network with $w(l)$ as the weight of link l.
5. Route the demand of D units from a to b along this shortest path, and update the residual capacities.

SMIRA, as defined above, clearly constitutes a general class of algorithms containing an unlimited number of particular instances (implementations). Here we consider two particular algorithmic instances, and investigate their performance. The first is called *minimum-interference bottleneck-link-avoidance* algorithm (MI-BLA), and is derived from the following choices:
– The set of critical paths is obtained using the K-widest-shortest-path under bottleneck-elimination procedure, with K equal to 6.
– All paths are considered to have the same weight, i.e. $v_{sd}^{(i)} = 1$.
– Links are valued according to the step function, i.e. $g_{sd}^{(i,l)} = \left\lfloor btl_{sd}^{(i)}/r(l) \right\rfloor$.
– Setting $a_{sd} = 1$ and $w = 2$, the weight of each link is then calculated by

$$w(l) = c(l) \left\{ 1 + 2 \sum_{(s,d)\in P\backslash(a,b)} \sum_{i=1}^{K} \sum_{l\in L_{sd}^{(i)}} \left\lfloor \frac{btl_{sd}^{(i)}}{r(l)} \right\rfloor \right\}.$$

The second is called *minimum-interference path avoidance* algorithm (MI-PA), and is derived from the following choices:
– The set of critical paths is obtained using the K-widest-shortest-path under bottleneck-elimination procedure, with K equal to 4.
– Paths are weighted according to the discounting function $v_{sd}^{(i)} = (K - i + 1)/K$.
– Links are valued inversely proportional, i.e. $g_{sd}^{(i,l)} = btl_{sd}^{(i)}/r(l)$.
– Setting $a_{sd} = 1$ and $w = 2$, the weight of each link is then calculated by

$$w(l) = c(l) \left\{ 1 + 2 \sum_{(s,d)\in P\backslash(a,b)} \sum_{i=1}^{K} \frac{K-i+1}{K} \sum_{l\in L_{sd}^{(i)}} \frac{btl_{sd}^{(i)}}{r(l)} \right\}.$$

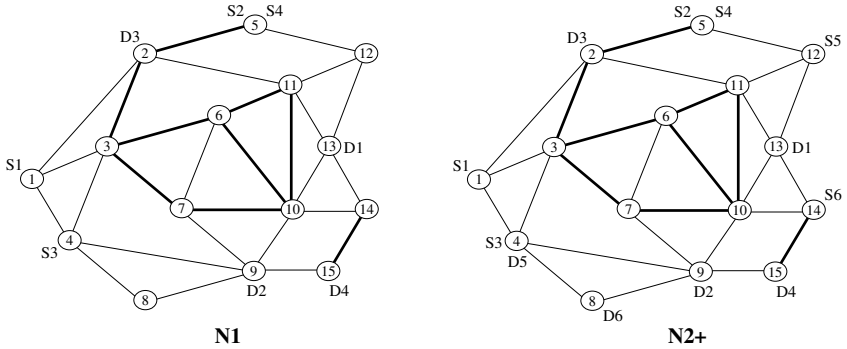

Fig. 1. Example networks N1 and N2+.

4 Numerical Results

In this section, we compare the performance of the two SMIRA-type algorithms MI-BLA and MI-PA with the shortest-path (SP), shortest-widest-path (SWP), widest-shortest-path (WSP), and, where results are available, with the S-MIRA and L-MIRA algorithms. The experiments are carried out using network topology N1 of [5], see Figure 1. Links are bi-directional with a capacity of 1200 units (thin lines) and 4800 units (thick lines).[1] Each link l is assigned a static cost $c(l)$ of one unit. The network contains the four ingress–egress pairs (S1→D1), (S2→D2), (S3→D3), and (S4→D4). Path requests are limited to those pairs only. We have chosen this network such that the performance results can be directly compared with those published in [5].

All experiments are conducted using "static" requests, i.e. the bandwidth allocated for a request is never freed again. Requests are selected randomly and are uniformly distributed among all ingress–egress pairs. In all experiments, 20 test runs were carried out, and the results shown are the mean values obtained. They have a 99% confidence interval not exceeding 1% of the mean values.

4.1 Experiment 1: Uniform Link Costs

In a first experiment, network N1 is loaded with 7000 requests. The bandwidth demand of each request is uniformly distributed in the range of 1 to 3 units (only integer values are used). Because the cost of the links in network N1 is set to 1, the SP algorithm is reduced to a minimum-hop algorithm.

The bandwidth of accepted requests of experiment 1 is shown in Figure 2. For each algorithm, the bandwidth increases with the number of requests until a saturation point is reached at which no more requests can be accommodated. The first performance measure we use is the bandwidth of successfully routed requests after the saturation point has been reached. The SP shows the weakest

[1] Owing to an error in the production of the final version of [5], links 2-5, 2-3, and 14-15 are erroneously shown as having a capacity of 1200 units. For the experiments described in [5], those links had a capacity of 4800 units.

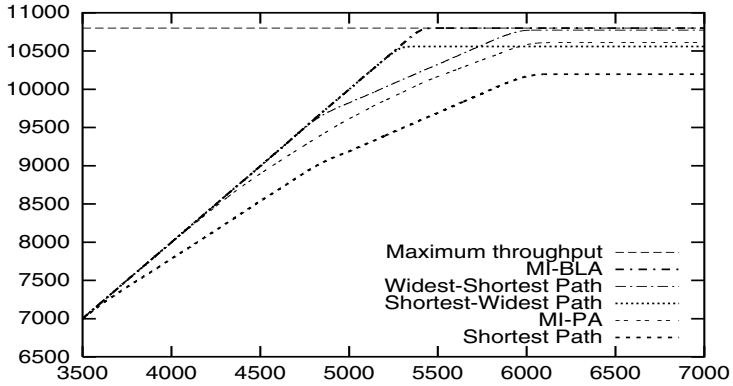

Fig. 2. Throughput of accepted requests using demands of 1 to 3 in N1.

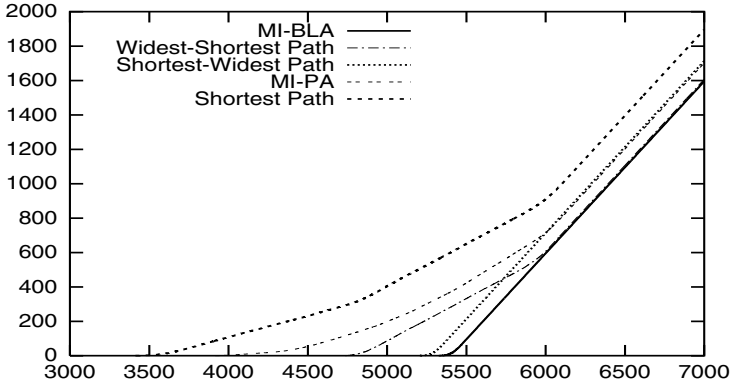

Fig. 3. Blocked requests using demands of 1 to 3 in N1.

performance, with a saturation point around 10200 bandwidth units. The best performance is shown by MI-BLA with 10800 units, followed very closely by WSP with 10770 units, and MI-PA and SWP with 10610 and 10550 units, respectively. Note that also the theoretical maximum is 10800 units. This maximum results from the solution of the multicommodity flow problem that maximizes the total flow of four commodities between the four ingress–egress pair. This is referred to as the *maximum throughput problem* [9]. Because requests are uniformly distributed among all ingress–egress pairs, we are also interested in a solution where the flow of each of the four commodities has the same value. This refers to the *maximum concurrent flow* variant of the multicommodity flow problem [9]. If the network is uniformly loaded, it is clear that a greedy routing algorithm cannot result in a flow that exceeds the maximum concurrent flow without rejecting any request. In our case, it turns out that the maximum throughput and the maximum concurrent flow have the same value of 10800. This implies that there is a solution where 2700 units can be transported between each ingress–egress pair, resulting in a total throughput of 10800 units. This maximum throughput is indicated by the "Maximum-throughput" line in Figure 2.

Table 1. Blocking rate per ingress–egress pair in N1.

Algorithm	(S1→D1)	(S2→D2)	(S3→D3)	(S4→D4)
MI-BLA	16.88%	16.81%	16.86%	16.82%
WSP	26.26%	8.42%	24.96%	8.43%
MI-PA	32.38%	8.81%	23.81%	8.79%
SWP	18.83%	18.64%	18.59%	18.59%
SP	45.25%	7.55%	25.54%	7.58%

Table 2. Number of blocked requests of total 5000 requests in N1.

Algorithm	Blocked requests			Algorithm	Blocked requests		
	Avg	Min	Max		Avg	Min	Max
Min-Hop	≈400	≈350	≈450	SP	404	353	448
WSP	≈340	≈310	≈380	WSP	86	28	151
S-MIRA	≈80	0	≈150	SWP	0	0	0
L-MIRA	0	0	0	MI-BLA	0	0	0

A second performance measure looks at the number of blocked requests. Figure 3 shows the number of blocked versus total requests. After 3450 requests, the SP algorithm starts to block some requests. MI-PA blocks after 3950 requests, WSP starts to block after 4750 requests, followed by SWP at 5230 and MI-BLA at 5350. From the above, it is clear that the blocking-free range strongly depends on the algorithm used. Note that although WSP starts to block quite early, it still achieves a throughput close to the theoretical maximum. Similarly, MI-PA has a short blocking-free range but achieves a higher total throughput than WSP does. This is due to the fact that the request-blocking rates of WSP and MI-PA differ significantly among the ingress–egress pairs. Table 1 shows the blocking rate per ingress–egress pair of various algorithms after 6500 requests have been processed, i.e. at a saturation point. MI-BLA and SWP almost achieve perfect fairness among the pairs, whereas WSP, MI-PA, and SP favor pairs 2 and 4.

Our results on the number of blocked requests are directly comparable with some of the results published in [5]. Figure 7 in [5] shows the number of blocked requests out of a total of 5000 requests for minimum-hop, WSP, S-MIRA, and L-MIRA. Table 2 compares the results presented in [5] (first four columns) with our results (columns 5 to 8). For all algorithms, the average, minimum, and maximum number of blocked requests are given.

We observe that Min-Hop closely matches our results of SP. Because uniform link costs have been used, SP actually computes minimum-hop paths. The results for WSP, however, do not match. In our experiment, WSP achieves a similar performance as S-MIRA does. Furthermore, we observe that L-MIRA achieves a perfect score with no blocked requests. In our experiment, we obtain the same result for both SWP and MI-BLA.

4.2 Experiment 2: Costs Inversely Proportional to Link Capacity

In the next experiment, we study the effect of static link costs on the performance. In network N1, all links have a cost of 1. We obtain network N2 by

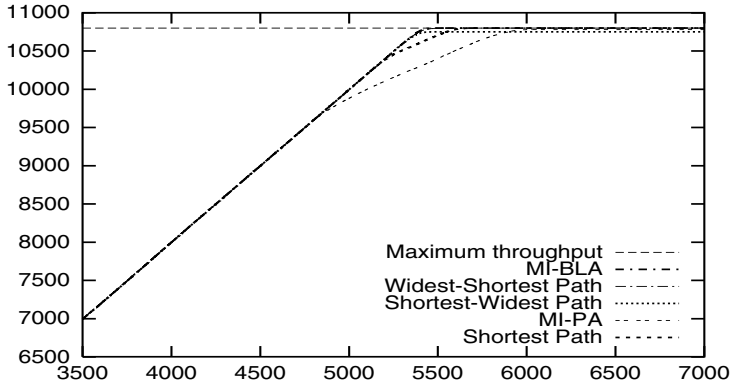

Fig. 4. Throughput of accepted requests using demands of 1 to 3 in N2.

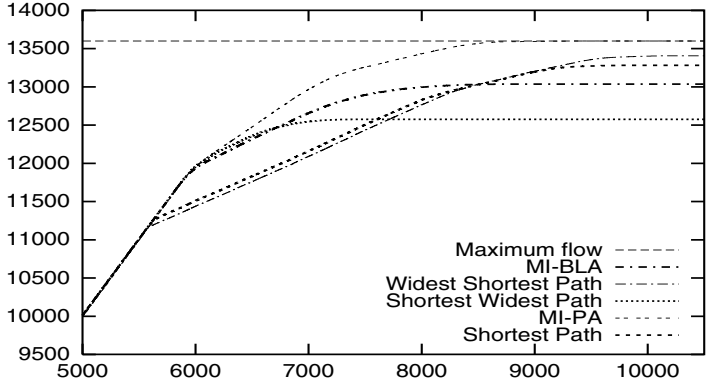

Fig. 5. Throughput of accepted requests using demands of 1 to 3 in N2+.

assigning different costs to the links. Following a common practice, we assign link costs inversely proportional to the link capacities. Links with capacity 1200 are assigned a cost of 4, and links of capacity 4800 are assigned a cost of 1. As shown in Figure 4, all algorithms perform almost equally well. The bandwidth routed by all algorithms is very close to the theoretical maximum of 10800 units. However, SP and in particular MI-PA achieve this maximum later than the other algorithms do.

4.3 Experiment 3: Additional Ingress–Egress Nodes

In a third experiment, we increase the possibility of "interference" by increasing the number of ingress–egress pairs. To obtain the example network N2+, two additional ingress–egress pairs have been added to N1, see Figure 1. A number of 11000 requests are issued, and as in the previous experiments, the requests are uniformly distributed among the six ingress–egress pairs.

Table 3. Maximum concurrent flow in N2+.

	(S1→D1)	(S2→D2)	(S3→D3)	(S4→D4)	(S5→D5)	(S6→D6)	Sum of flows
Step 1	2000	2000	2000	2000	2000	2000	12000
Step 2	2400	2000	2400	2400	2000	2000	13200
Step 3	2400	2000	2800	2400	2000	2000	13600

As shown in Figure 5, the best performance is achieved by MI-PA, reaching a total throughput of close to 13600 units. WSP and SP exhibit a very similar behavior: both start to block requests early, but are able to successfully route a total of 13400 and 13300 units, respectively. MI-BLA, on the other hand, starts to block later, but saturates earlier, at 13000 units. WSP is the least successful strategy in this environment, it reaches its saturation point already at 12600.

To compute the theoretical maximum performance of greedy algorithms, we resort to the multicommodity flow problem that maximizes the flow of six commodities corresponding to the ingress–egress pairs. In this case, it turns out that the maximum concurrent flow and the maximum throughput of the multicommodity flow problems do not coincide. In a first iteration, we find that a maximum of 2000 units of flow can be transported between each pair, resulting in a maximum concurrent flow of 12000 units. The flow can no longer be increased because some pairs are saturated. In our case it turns out that there still is residual bandwidth left between pairs (S1→D1), (S3→D3) and (S4→D4). If we compute the maximum concurrent flow in the residual network for the unsaturated pairs, we find that these pairs support another 400 units of flow. In a third step, we find that pair (S3→D3) supports another additional 400 units of flow. With this three-step approach, we obtain the maximum throughput as 13600 units. Table 3 summarizes the three maximum concurrent flow computations.

Table 3 also defines how the optimum algorithm works in the settings of experiment 3. Requests for pairs (S2→D2), (S5→D5) and (S6→D6) are blocked after 2000 units of bandwidth have been routed over those pairs. Next, request for pairs (S1→D1) and (S4→D4) are blocked after 2400 units of bandwidth. Finally, requests for pair (S3→D3) are blocked. At this point, an optimum algorithm reaches its saturation point, with 13600 units of bandwidth routed in total. For an average request size of 2, the saturation point is expected to be reached at 8400 requests.

MI-PA achieves near-optimum performance with respect to the total throughput. Figure 6 shows that also MI-PA is very close to the optimum solution with respect to the request-acceptance rate of individual pairs. MI-PA slightly over-allocates requests for pair 3 at the expense of pair 6. The request-acceptance rates of individual pairs differ significantly for WSP and SP. Compared with the optimum solution (shown on the left), both WSP and SP over-allocate requests for pairs 3 and 4, while under-allocating requests for other pairs. MI-BLA and SWP, on the other hand, show a greater fairness among the pairs.

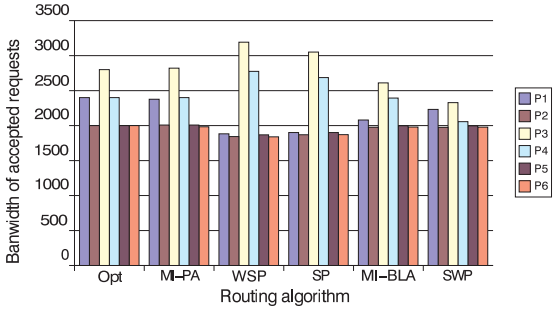

Fig. 6. Bandwidth of accepted requests per ingress–egress pair.

5 Conclusions

Here we have addressed the issue of on-line path selection for bandwidth-guaranteed requests. We have presented a new class of minimum-interference routing algorithms called "simple minimum-interference routing algorithms" (SMIRA), designed for a reduced computational complexity compared with the existing MIRA maximum-flow approach. Two typical algorithms, called MI-BLA and MI-PA, belonging to this class were introduced, and their efficiency in terms of the throughput of accepted requests and blocking-free range, as well as their fairness were assessed by means of simulation. The results obtained in the topologies considered demonstrate that these algorithms can achieve a similar optimum performance as the earlier MIRA algorithm, however, at reduced computational complexity. Comparisons with the performance of some of the established routing algorithms revealed that employment of MI-BLA and MI-PA in networks with a high degree of interference improves the performance compared with that of the shortest-path, widest-shortest-path, and shortest-widest-path algorithms. Furthermore, our algorithms exhibit a higher degree of fairness among the ingress–egress node pairs. An investigation and assessment of how the algorithms proposed perform in dynamic environments is a significant area of future work. A more systematic approach for determining the optimum algorithmic instance within the SMIRA algorithm is also a topic for further investigation.

References

1. Rosen, E., Viswanathan, A., Callon, R.: Multiprotocol Label Switching Architecture. RFC 3031 (January 2001).
2. The ATM Forum: Private Network-Network Interface Specification Version 1.0. Specification Number af-pnni-0055.000 (March 1996).
3. Ma, Q., Steenkiste, P.: On Path Selection for Traffic with Bandwidth Guarantees. In: Proc. IEEE Int'l Conf. on Network Protocols, Atlanta, GA (1997) 191-202.
4. Gawlick, R., Kalmanek, C., Ramakrishnan, K. G.: On-line Routing for Permanent Virtual Circuits. In: Proc. IEEE INFOCOM '95, Boston, MA, Vol. **1** (1998) 278-288.

5. Kar, K., Kodialam, M., Lakshman, T. V.: Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications. IEEE J. Sel. Areas Commun. **18** (2000) 2566-2579.
6. Gibbens, R. J., Kelly, F. P., Key, P. B.: Dynamic Alternative Routing – Modelling and Behavior. In: Proc. 12th Int'l Teletraffic Congress, Turin, Italy (1988) 1019-1025.
7. Suri, S., Waldvogel, M., Warkhede, P. R.: Profile-Based Routing: A New Framework for MPLS Traffic Engineering. In: Boavida, F., Ed., Quality of Future Internet Services, LNCS **2156** (Springer, Berlin, 2001).
8. Dijkstra, E. W.: A Note on Two Problems in Connexion with Graphs. Numerische Mathematik **1** (1959) 269-271.
9. Aumann, Y., Rabani, Y.: An $O(\log k)$ Approximate Min-Cut Max-Flow Theorem and Approximation Algorithm. SIAM J. Comput. **27** (1998) 291-301.