Available at www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

ELSEVIER

Computer Networks 42 (2003) 503–520

COMPUTER NETWORKS

www.elsevier.com/locate/comnet

Bandwidth–delay constrained routing algorithms ☆

Yi Yang [1], Lei Zhang, Jogesh K. Muppala, Samuel T. Chanson *

Department of Computer Science, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong

Received 22 April 2002; received in revised form 19 October 2002; accepted 26 January 2003

Responsible Editor: M.U. Çaglayan

Abstract

The development of technologies like Multi-Protocol Label Switching (MPLS) and Differentiated Services has laid the foundation for Internet to support multimedia applications like Voice over IP. Although much work has been done on laying MPLS paths to optimize performance, most has focused on satisfying bandwidth requirements. Relatively little research has been done on laying paths with both bandwidth and delay constraints. In this paper, we present bandwidth–delay constrained routing algorithms that use knowledge of the ingress–egress node pairs in the network in reducing the rejection rates for setting up new paths. Simulation is used to evaluate the algorithms and compare their performance against some existing algorithms for bandwidth constraints that have been modified to handle delay constraints. The results show that the proposed algorithms outperform all others under a wide range of workload, topology and system parameters.

Keywords: Quality of service routing; Bandwidth–delay constrained routing; Traffic engineering; MPLS; Maximum flow

1. Introduction

New mechanisms for supporting quality of service (QoS) on the Internet, including Differentiated Services (Diffserv) [3], and Multi-Protocol Label Switching (MPLS) [2] have given hope for transforming the best-effort based Internet into an infrastructure capable of supporting delay sensitive applications like Voice over IP (VoIP) [16]. From the perspective of this paper, VoIP is representative of a class of applications that require support for setting up bandwidth–delay constrained paths through the network.

Given a network path, the routers where the path originates and terminates are known as the ingress and egress routers respectively. In network models like MPLS, the ingress and egress routers where a requested path can potentially originate and terminate are known, since they are all network edge routers. This knowledge of ingress–egress pairs is useful since it restricts to some extent the links through which traffic will flow.

☆ This paper is an extension of our earlier work [23] that appeared in International Conference on Network Protocols (ICNP) 2001. The MDWCRA algorithm is enhanced with an average performance improvement of 20%.

* Corresponding author. Tel.: +852-23586982; fax: +852-23581477.

E-mail address: chanson@cs.ust.hk (S.T. Chanson).

[1] Present address: Department of Computer Science, University of California Los Angeles, Los Angeles, CA, USA.

1389-1286/03/$ - see front matter
doi:10.1016/S1389-1286(03)00199-3

However, most existing bandwidth–delay constrained routing algorithms [6,15,22] assume that the only dynamic information available is the link residual bandwidth and delay. The availability of quasi-static information such as the locations of the ingress–egress nodes in the network was first exploited only recently [13,14] to reduce the number of rejected connection setup requests. The path setup requests are restricted to take place only among specific ingress–egress pairs. Their focus was on routing in the context of setting up bandwidth guaranteed MPLS paths. Not much research has been done on path selection algorithms using knowledge of ingress–egress nodes and considering both bandwidth and delay criteria.

In this paper, we concentrate on the specific problem of designing bandwidth–delay constrained algorithms taking into account knowledge of the ingress–egress node pairs. The path setup requests arrive one by one, and future demands are unknown. Our algorithms are based on computing the *delay-weighted capacity* (DWC) for each ingress–egress pair. We then identify *critical links* as those links whose inclusion in a path will cause the DWC of several ingress–egress pairs to decrease. The algorithms avoid using the critical links as far as possible by assigning large weights to them as a function of their criticality. We use an extended Dijkstra (EDSP) or Bellman–Ford (EBF) algorithm for selecting the path with the least weight.

We compared the performance of our algorithms with others derived by modifying some bandwidth constrained routing algorithms to support both bandwidth and delay constraints. The *call-blocking ratio,* defined as the ratio of the number of rejected requests to the total number of requests, is used as the performance metric. We conducted simulation experiments with two different network topologies. The results show that our algorithms, named maximum DWC routing algorithms (MDWCRA), including the modified MDWCRA (M-MDWCRA) which considers more paths and critical links, outperform the other algorithms under a wide range of workload and system parameters. The performance gain is achieved without a significant increase in overhead.

In the following sections we first give a brief discussion on related work in Section 2. We then define the problem to be studied in Section 3. Key ideas of our approach are presented in Section 4, and new algorithms are proposed in Section 5. Details of the network configurations and the experimental setup used in evaluating the algorithms are presented in Section 6. The performance results are discussed in Section 7. Finally Section 8 concludes the paper with some suggestions for future work.

2. Related work

The QoS routing problem can be viewed as composed of four basic classes, namely, link-optimization routing, link-constrained routing, path-optimization routing and path-constrained routing [7]. Link-optimization routing and link-constrained routing are defined for concave QoS metrics like bandwidth and buffer space. Path-optimization routing and path-constrained routing are defined for additive and multiplicative metrics like delay and delay jitter. Composite routing problems can be derived from these four basic routing problems.

The bandwidth-constrained routing problem, which belongs to link-constrained routing, is the most addressed problem [1,6,12,14,16]. Two commonly used algorithms are the minimum hop (Min-Hop) [12] and the widest available path first (WAPF) [16]. Min-Hop selects the path with the least number of feasible links. WAPF chooses the path with the maximum available capacity from the source to the destination.

The bandwidth–delay constrained routing problem, which is the focus of this paper, belongs to the category of link-constrained path-constrained routing that is solvable in polynomial time. Typically, the routing is performed in two steps. All nodes and links with insufficient resources to satisfy the constraints are first eliminated from the network. Then a shortest path algorithm like Dijkstra or Bellman–Ford algorithm is used to find a feasible path in the remaining graph. In the Wang and Crowcroft algorithm [22] the links with available bandwidth less than the bandwidth re-

quirement are first eliminated. Then the shortest feasible path from the source to the destination in terms of delay is chosen. A distributed routing algorithm called Shortest Widest Path (SWP) [22] finds a feasible widest available path between the ingress and egress nodes. If multiple paths exist, the path with the minimum delay, called the SWP, is selected.

The Minimum Interference Routing algorithm (MIRA) [13,14] exploits the knowledge of ingress–egress pairs in finding a feasible path. The idea is that a newly routed connection should follow a path that does not *interfere too much* with a path that may be critical to satisfy a future demand. A *Critical* link is identified as one with the property that if a path is routed through it, the maxflow value of one or more ingress–egress pairs decreases. The algorithm aims to avoid these critical links while making path selection. It exhibits very good performance compared to other routing algorithms. However it concentrates only on setting up bandwidth guaranteed paths. In [13] the possibility of incorporating various policies, hop-count and delay constraints within the bandwidth-routing framework is discussed. This is done by translating other constraints into effective bandwidth requirements. Specifically for delay constraints they suggest that the problem becomes a constrained shortest path problem which is NP-hard. They suggest using a pseudo-polynomial time algorithm or heuristic approach to solve the problem.

Inspired by MIRA, Profile-Based Routing (PBR) [20,21] was proposed in 2001 to deal with the dynamic routing problem of bandwidth-guaranteed flows. PBR uses a "traffic profile" of the network, obtained by measurements or service-level agreements, as a rough predictor of the future traffic distribution. The "profile" is used to solve a *multi-commodity network flow* problem, whose output is used to guide online path-selection as well as to impose admission control. PBR improves MIRA on the number of accepted requests. However, PBR works with bandwidth only and, unlike most of the other existing algorithms, it cannot be easily extended to deal with delay. When only bandwidth is considered, the average bandwidth requirement (or the maximum, the aggregation, etc.) over a certain period of time can be used as the "traffic profile". None of this seems applicable to obtain "traffic profile" for delay. Therefore, this algorithm is not included in our simulation comparison since it is not meaningful to compare algorithms designed for different purposes.

Both path-constrained path-optimization routing and multi-path-constrained routing are NP-complete. Among these classes, delay-cost-constrained routing and delay-constrained least-cost routing have received the most attention [6,17,18]. Most algorithms transform the NP-complete problem into a problem that can be solved in polynomial time. We will not investigate these problems further in this paper.

3. Problem definition

The network consists of n routers. A subset of these routers is assumed to be ingress–egress routers between which paths can be set up. We assume that the ingress–egress nodes are known and change infrequently. Each link in the network has two properties: residual bandwidth and delay. The residual bandwidth is defined as the difference between the link bandwidth and the sum of the bandwidths of all the paths already assigned to that link. The delay of a link consists of the link propagation delay and the queuing delay at the starting node. We use either a source [7] or server-based routing strategy [1]. Both strategies are simple and can guarantee loop-free routes. In source routing, each router maintains a complete and up-to-date global state. A path setup request arrives at an ingress router that locally computes an explicit route for the request. In server-based routing, a single entity called route server keeps the complete link state topology database and is responsible for finding a feasible path. A request either directly reaches the route server or first arrives at an ingress router that forwards the request to the route server. The route server generates an explicit route and sends back to the ingress router. The ingress router then sets up the path to the egress and reserve resource on each link along the path. For computing the explicit route the ingress

router or the route server needs to know the current network topology, link residual bandwidth and delay. We assume that this information is either known or that a link state routing protocol is used to acquire the information.

Some of the notations to be used in this paper are defined below. The network is modeled as a directed graph $G(V, E, P)$ where V is the set of nodes (routers), E is the set of edges representing directed communication links between the nodes in V. Let n represent the number of routers and m the number of links in the network. Each link $l_{ij} \in E$ is associated with a vector (b_{ij}, d_{ij}), where b_{ij} is the residual bandwidth and d_{ij} is the delay of link l_{ij}. P is considered as the set of potential ingress–egress pairs. We denote a generic element of this set P by (s, t). All path setup requests are assumed to occur between these pairs. Let p denote the total number of pairs. The setup request for path i is defined by a quadruple (s_i, t_i, B_i, D_i), where s_i specifies the ingress router, t_i specifies the egress router, $B_i \in R_0^+$ specifies the amount of bandwidth required and $D_i \in R_0^+$ specifies the delay requirement.

We assume that path setup requests arrive one at a time and there is no prior knowledge of future requests. The objective is to determine a feasible path for each request. We use the *call blocking ratio* as the performance metric to compare the different algorithms:

$$\text{call blocking ratio} = \frac{\text{number of requests rejected}}{\text{total number of requests}}.$$

The optimization goal is to minimize the *call blocking ratio*, which in turn will maximize the number of requests accepted into the network.

4. Key principles

In this section, we present the key ideas used in our routing algorithms. A request can be accepted if sufficient resources are available to satisfy its bandwidth and delay requirements. Therefore we should conserve as much resources as possible that would be *critical* to meet future demands. We use the knowledge of the network's ingress and egress routers to determine the criticality of the links.

4.1. Delay-weighted capacity

Consider an ingress–egress pair (s, t). Let us imagine a path between s and t as a kind of *machine* to accommodate requests with bandwidth and delay constraints. We can measure the *power* of the machine by the end-to-end delay of that path. The smaller the value, the more powerful the machine since the machine can satisfy requests with tight delay requirements. The number of requests the machine can accommodate, called the *capacity* of the machine, is measured by the bandwidth of the path. It is easy to see that the most *powerful machine* for the pair (s, t) is the least delay path between s and t. We can then remove from the network all links used by the most powerful machine, i.e., the links belonging to the least delay path. The second most powerful machine for the pair (s, t) is the least delay path computed from the remaining graph. Repeating this process until no path exists between s and t, we get a set of least delay paths represented by $\mathrm{LP}_{st} = \{\mathrm{LP}_{st}^1, \ldots, \mathrm{LP}_{st}^i, \ldots, \mathrm{LP}_{st}^{k_{st}}\}$. LP_{st}^i is the least delay path computed using the graph where the links belonging to $\mathrm{LP}_{st}^1, \ldots, \mathrm{LP}_{st}^{i-1}$ are eliminated. Let B_{st}^i denote the residual bandwidth and D_{st}^i the end-to-end delay of LP_{st}^i. The total number of paths in LP_{st} is k_{st}.

An example is shown in Fig. 1, with node 1 and node 6 as the only ingress–egress pair. In this figure, the first least delay path between s ($s = 1$) and t ($t = 6$) is $P_1(1, 2, 3, 6)$ with bandwidth 1 and delay 3. Path P_1 is abstracted as a dotted link from s to t as shown in Fig. 1(b). The second least delay path $P_2(1, 4, 5, 6)$ with bandwidth 1 and delay 6 is then calculated in Fig. 1(b) and abstracted as a link in Fig. 1(c). The process repeats until no path can be found as shown in Fig. 1(c), where the network is abstracted as a set of dotted links between s and t representing the set of least delay paths computed.

We consider maximizing the sum of machine capacity for each ingress–egress pair (s, t). Each machine capacity is weighted by the power of that machine which indicates the importance of the

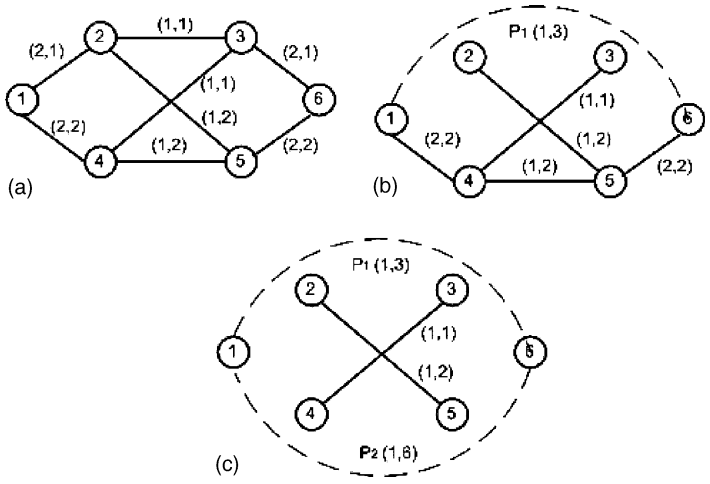


Fig. 1. Illustrative example. Vector associated with each link is (bandwidth, delay).

capacity. We define the DWC for each pair (s,t) below.

Definition 1. The DWC of ingress–egress pair (s,t) is defined as a weighted sum of the bandwidth of the paths in the set $\mathrm{LP}_{st} = \{\mathrm{LP}_{st}^1, \ldots, \mathrm{LP}_{st}^i, \ldots, \mathrm{LP}_{st}^{k_{st}}\}$. The weights are inversely proportional to the end-to-end delay values of these paths:

$$\mathrm{DWC}_{st} = \sum_{\mathrm{LP}_{st}^i \in \mathrm{LP}_{st}} \frac{B_{st}^i}{D_{st}^i}.$$

Following this definition, the DWC value of ingress–egress pair (1, 6) is $(1/3) + (1/6) = 0.5$.

4.2. Critical link

The bandwidth of a path is determined by the minimum link bandwidth along that path. Links that determine the path bandwidth are considered as *bottleneck* links for that path. If we route a request on a bottleneck link of any least delay path in LP_{st}, the DWC value of the pair (s,t) decreases. Such a link is defined as a *critical link* for (s,t). It has the property that whenever a path is routed over it, the DWC value of one or more ingress–egress pairs decreases. We represent the set of critical links for (s,t) by $C_{st} = \{C_{st}^1, \ldots, C_{st}^i, \ldots, C_{st}^{k_{st}}\}$, where C_{st}^i consists of all the bottleneck links for the least delay path LP_{st}^i. In the example given in Fig. 1, the critical links are the bottleneck links of P_1 and P_2, which are link (2, 3) and link (4, 5) respectively.

4.3. Path selection

It is important to note the dynamic nature of DWC, since it is recomputed after each path allocation. It is clear we should avoid routing paths on the critical links as much as possible. We do this by assigning weights to critical links that are an increasing function of their *criticality*. An EDSP or EBF algorithm is used to compute the least weight path.

5. Routing algorithms

We wish to maximize the weighted sum of DWC of each ingress–egress pair after satisfying the current request. We achieve this by determining appropriate weights for the links in the network and route the request along the least weight path. The weights are an increasing function of the criticality of the links. Therefore the problem of computing the weights of the links is reduced to one of determining the set of critical links for all ingress–egress pairs. This can be solved as an iterative process of calculating LP_{st} for each pair (s,t). We can use Dijkstra algorithm to compute

the least delay path LP^i_{st} in each round in the iteration, which takes $O(n \log n)$. Critical links for LP^i_{st} can be found in $O(n)$ since LP^i_{st} contains at most $(n-1)$ links. For each ingress–egress pair, the iteration takes at most $O(m)$ rounds since at least one link is eliminated from the graph in each round. We can further reduce the number of rounds to $O(1)$ as follows. We know from computational geometry [4] that the average degree d of a node in a planar graph is 6. We know that any path between two nodes includes a link originating from the source and an incoming link to the destination. Each time we eliminate the links along the least delay path between an ingress–egress pair, we decrease the outgoing degree of the ingress node and the incoming degree of the egress node by 1. On average, after 6 rounds, no path can be found between that pair of ingress–egress nodes. Therefore for each ingress–egress pair, it takes $O(n \log n + n)$ to compute LP_{st} and find all critical links. Since there are a total of p ingress–egress pairs, the complexity is $O(p(n \log n + n)) = O(np \log n)$.

Once all the critical links are determined, we assign weights to the critical links and route the request along the least weight path. Before doing so, we first eliminate all links with residual bandwidth less than the bandwidth requirement so as to ensure that any path computed in the remaining graph will satisfy the bandwidth constraint. To guarantee that the delay constraint is also satisfied, we use an EDSP or EBF algorithm [5] to compute the least weight path. Both of these algorithms can solve the two-additive-constrained routing problem. Therefore we can ensure the least weight path computed by EDSP or EBF is feasible in terms of delay. We call such a path the delay-constrained least-weight path. The complexity of ESDP is $O(x^2n^2)$ and the complexity of EBF is $O(xmn)$ where x is a positive integer.

We assume the current request is between routers a and b with demands of B units of bandwidth and D units of end-to-end delay. At this point other requests may already have been routed and the residual capacities of the links have been updated to reflect these allocations. We denote this graph with G. The routing algorithm is detailed below, where α^i_{st} is a property associated with least delay path LP^i_{st}. We shall discuss how to set the value of α^i_{st} later.

5.1. Maximum delay-weighted capacity routing algorithm

1. Initialize all link weights to 0.
2. For each ingress–egress pair $(s,t) \in P$, compute the DWC value for it.
   - 2.1. Initialize a working graph, which is the same as the current residual graph G. $i = 1$.
   - 2.2. While there is still path from s to t in the working graph,
   - 2.3. Find the ith least delay path LP^i_{st} for pair (s,t).
   - 2.4. Label the bottleneck links of the least delay path as critical, and add them to C_{st}.
   - 2.5. For each of the critical links identified, update the link weight $w_l = w_l + \alpha^i_{(s,t)}$.
   - 2.6. Delete all links belonging to LP^i_{st} in the working graph.
   - 2.7. $i = i + 1$. Go to 2.2.
3. Eliminate all links in G that have residual bandwidth less than B to form a reduced network.
4. Using the EDSP or EBF algorithm, compute the delay-constrained least-weight path in the reduced network using w_l as the weight on link l.
5. Route the request from a to b along this delay-constrained least-weight path and update the residual capacities of the network.

In the above algorithm, the link weights are actually computed according to the definition of $w_l = \sum_{(s,t):l \in C^i_{st}} \alpha^i_{st} \ \forall l \in E$. By varying the value of α^i_{st}, different definitions of link weight can be obtained as follows:

1. $w_l = \sum_{(s,t):l \in C^i_{st}} 1$, with $\alpha^i_{st} = 1$.
   The weight of link l represents the number of ingress–egress pairs for which link l is critical.
2. $w_l = \sum_{(s,t):l \in C^i_{st}} 1/D^i_{st}$, with $\alpha^i_{st} = 1/D^i_{st}$.
   The link weight is inversely proportional to the end-to-end delay value of the least delay path.
3. $w_l = \sum_{(s,t):l \in C^i_{st}} 1/(B^i_{st} \times D^i_{st})$, with $\alpha^i_{st} = 1/(B^i_{st} \times D^i_{st})$.

The link weight is inversely proportional to the product of the bandwidth and the end-to-end delay of the least delay path.

Now we analyze the time complexity of MDWCRA. As discussed above, Step 2 takes $O(np\log n)$. Step 3 costs $O(m)$. Step 4 takes the same amount of time as the EDSP or EBF algorithm. The total time complexity is therefore $O(np\log n + m + x^2n^2) = O(np\log n + x^2n^2)$ by ESDP or $O(np\log n + m + xmn) = O(np\log n + xmn)$ by EBF.

5.2. Modified maximum delay-weighted capacity routing algorithm

In MDWCRA, the least delay path is found and all the links along the path are eliminated from the working graph in each round before finding the next least delay path. Only the bottleneck links of these least delay paths are considered critical. However, other links along the paths that have just been eliminated may become critical without being protected by the algorithm. Using the same example in Fig. 1, $P_1(1,2,3,6)$ and $P_2(1,4,5,6)$ will be identified by the MDWCRA algorithm, with their corresponding critical links (2, 3) and (4, 5) protected. The bottleneck links of $P_3(1,2,5,6)$ and $P_4(1,4,3,6)$ are not protected.

A modified version of the MDWCRA algorithm is proposed to address this problem. In the process of assigning weights to the links (Step 2.6 in the algorithm), instead of eliminating all links along the least delay path, only the bottleneck link is deleted from the working graph in each round. This allows more paths in the graph to be considered in selecting the next least delay path and protects more links which may become critical. Fig. 2 shows how the modified algorithm works using the graph in Fig. 1. P_1, P_4, P_3, and P_2 are found to be the least delay paths in each round with their corresponding bottleneck links identified as critical and deleted from the graph. The critical links are assigned weights and the least weight path is used to route the request. Using the first definition of link weights, link (2, 3), link (4, 3), link (2, 5), and link (4, 5) will obtain weights of 1/3, 1/4, 1/5, and 1/6 respectively.

In the worst case, the number of paths searched by M-MDWCRA is equal to the number of links

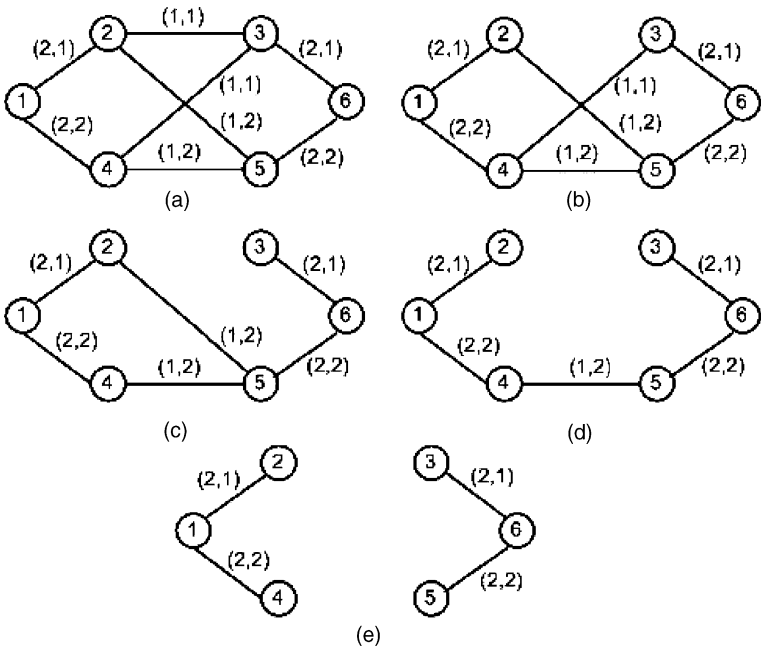


Fig. 2. Illustrative example of the M-MDWCRA algorithm. Vector associated with each link is (bandwidth, delay).

in the graph. Since the average degree of node in a planar graph never exceeds six [4], the number of links is bounded by $3n$, where n is the number of nodes. Therefore, the time complexity of M-MDWCRA is $O(n^2p\log n + x^2n^2)$ by ESDP or $O(n^2p\log n + xmn)$ by EBF.

Simulation results show that the modified version can improve the performance of MDWCRA by an average of 20%.

6. Network configuration

Two different network topologies were used to compare the different routing algorithms. The first topology, adopted from [14], is called the MIRA topology and consists of 15 nodes as shown in Fig. 3(a). All the links are bidirectional. There are two different kinds of links in the network: The thin links have a capacity of 12 units and the thick links have a capacity of 48 units. A subset of the nodes in the network acts as the ingress–egress pairs. In the MIRA topology, four ingress–egress pairs are considered, which are (0, 12), (4, 8), (3, 1), and (4, 14).

The second topology, adopted from [5], expands the major circuits in ANSNET by inserting additional links to increase the connectivity. This topology, called expanded ANSNET topology, consists of 32 nodes and is presented in Fig. 3(b). Each link has a capacity of 12 units. Five pairs in this topology are considered as ingress–egress pairs, which are (1, 29), (18, 6), (4, 23), (7, 31), and (21, 17).

An important part of the simulation environment is the link delay distribution. Note that link delay consists of queuing delay as well as propagation delay. The common approaches used in the literature for modeling link delay are summarized below:

(i) Link delay values are assumed to be uniformly distributed within some range [5,11]. A variation of this method groups links into short local links (1–5 ms), longer local links (5–8 ms), and continental links (20–30 ms) [10].
(ii) Link delay values are generated according to measured network traffic statistics. For example, Fig. 4 shows the link delay distribution measured in the NSFNET T1 backbone [8].

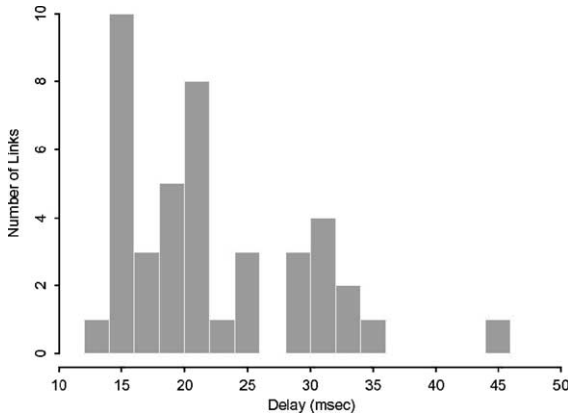
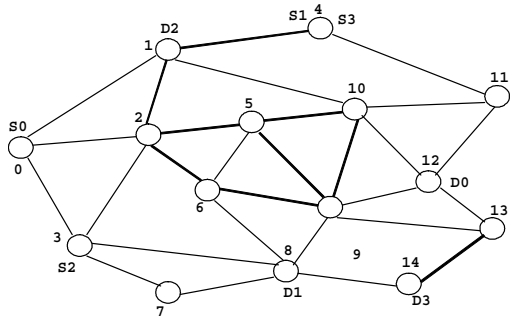
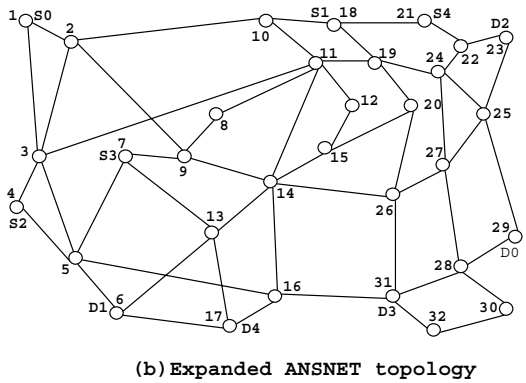


Fig. 4. Distributions of one-way median delays across NSFNET T1 backbone.



(a) MIRA topology



(b) Expanded ANSNET topology

Fig. 3. Network topologies.

(iii) Nodes in the network are assumed to be scattered across the US and propagation delay along these backbone trunks is taken to be the dominant factor of the total link delay. Therefore, link delay values are calculated by dividing node physical distances by the signal speed in the fiber. This approach is used in [18,19].

We have performed simulations using each of these three approaches and obtained similar conclusions. In this paper, only the results of the third link delay generating method is presented in detail for lack of space. Moreover, this method seems more practical for the future MPLS backbone. [2] The nodes are placed in a 3000×2400 km^2 rectangle, roughly the size of the USA, and the propagation speed through the links is two thirds of the speed of light. Propagation delays are used as the link delays in the simulation. Some simulation results using the first two link delay generating approaches are given in the Appendix A for reference.

We also examine the performance of the different algorithms in both the MIRA topology and the expanded ANSNET topology with unspecified ingress–egress pairs, i.e., any pair can be the source–destination pair.

We consider both even and uneven distributed traffic load. For an even load, path setup requests are generated randomly between any source–destination pair. The pair is randomly picked from all possible ingress–egress pairs, with each pair having the same probability of being selected. For an uneven load, a large percentage of the traffic is distributed between a selected subset of ingress–egress pairs. These heavily-loaded pairs are explicitly specified in the MIRA topology and the expanded ANSNET topology with specified ingress–egress nodes. In the case where ingress–egress nodes are not specified, the heavily-loaded pairs are randomly selected from all possible pairs.

[2] In fact, propagation delay is becoming the dominant factor as the Internet moves to higher and higher speed with large buffer size, especially for the backbones [9].

The bandwidth requirement of a request is uniformly distributed between 1 and 5 units. The delay requirement is generated from different ranges, viz., [25–35 ms], [45–55 ms], [65–75 ms], [125–140 ms] and [150–165 ms]. The smaller the delay range, the tighter the delay constraint of a request. Within each range, a delay requirement is uniformly distributed.

7. Performance results

We considered the following algorithms:

1. The Wang and Crowcroft algorithm with complexity of $O(n \log n + m)$ [22].
2. The delay-constrained Min-Hop (DC Min-Hop) algorithm with complexity of $O(x^2n^2)$ or $O(xmn)$.
3. The delay-constrained WAPF (DC WAPF) algorithm with complexity of $O(x^2n^2)$ or $O(xmn)$.
4. The delay-constrained MIRA (DC MIRA) algorithm with complexity of $O(pn^2\sqrt{m} + x^2n^2)$ or $O(pn^2\sqrt{m} + xmn)$.
5. MDWCRA algorithm with complexity of $O(np \log n + x^2n^2)$ or $O(np \log n + xmn)$.
6. M-MDWCRA algorithm with complexity of $O(n^2p \log n + x^2n^2)$ or $O(n^2p \log n + xmn)$.

The original Min-Hop [12], WAPF [16] and MIRA [14] algorithms deal with the case of setting up bandwidth-guaranteed connections between given source and destination pairs. These three algorithms utilize the well known EDSP or EBF to calculate the feasible path. We modify them by using the EDSP or EBF algorithm to ensure that the path computed by the extended Min-Hop, WAPF and MIRA algorithms satisfy the delay constraint. We call the three extended algorithms the delay-constrained Min-Hop (DC Min-Hop), DC WAPF and DC MIRA algorithms respectively. The complexity of DC Min-Hop and DC WAPF is determined by the complexity of EDSP ($O(x^2n^2)$) or EBF ($O(xmn)$). Using knowledge of ingress–egress nodes introduces additional complexity in the order of $O(pn^2\sqrt{m})$ in DC MIRA and $O(np \log n)$ in MDWCRA. It is clear that $O(np \log n)$ is much smaller than $O(pn^2\sqrt{m})$. As

Table 1
Number of critical links identified for each request

Parameters			MDWCRA	M-MDWCRA	Ratio
Specified ingress–egress pairs	MIRA topology	Even traffic	11.19	18.57	1.66
		Uneven traffic	10.88	18.19	1.67
	Expanded ANSNET topology	Even traffic	11.05	22.68	2.05
		Uneven traffic	12.22	27.93	2.29
Non-specified ingress–egress pairs	MIRA topology		56	56	1
	Expanded ANSNET topology		108	108	1

Simulations are based on the average of multiple runs with 600 requests.

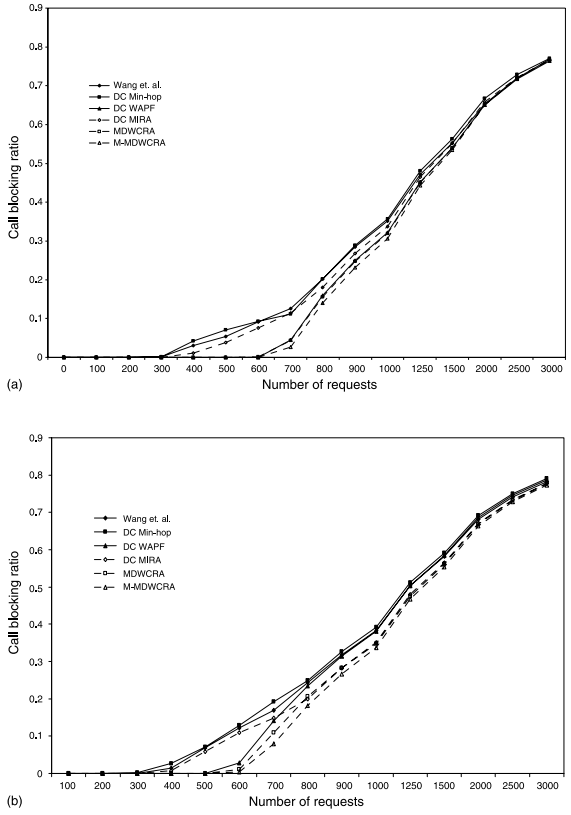


Fig. 5. Call blocking ratio as function of the number of requests. MIRA topology with specified ingress–egress nodes, $B \in [1\text{–}5]$, $D \in [25\text{–}35$ ms]: (a) evenly distributed traffic, (b) unevenly distributed traffic.

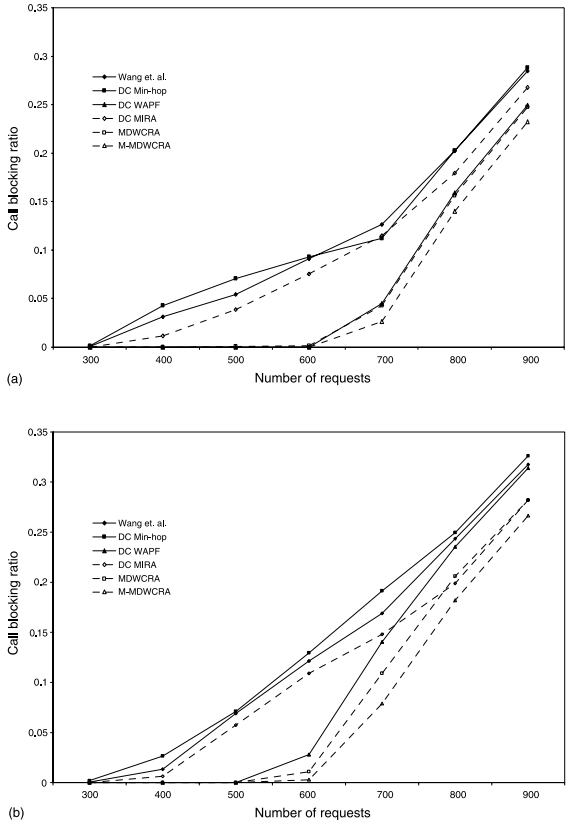


Fig. 6. Enlarged parts of Fig. 5: (a) evenly distributed traffic, (b) unevenly distributed traffic.

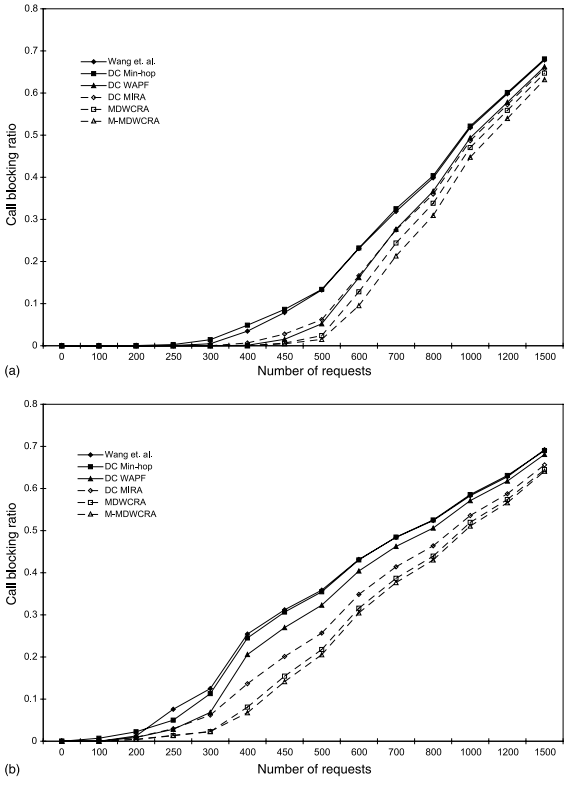


Fig. 7. Call blocking ratio as function of the number of requests. Expanded ANSNET topology with specified ingress–egress nodes, $B \in [1\text{–}5]$, $D \in [30\text{–}40$ ms]: (a) evenly distributed traffic, (b) unevenly distributed traffic.

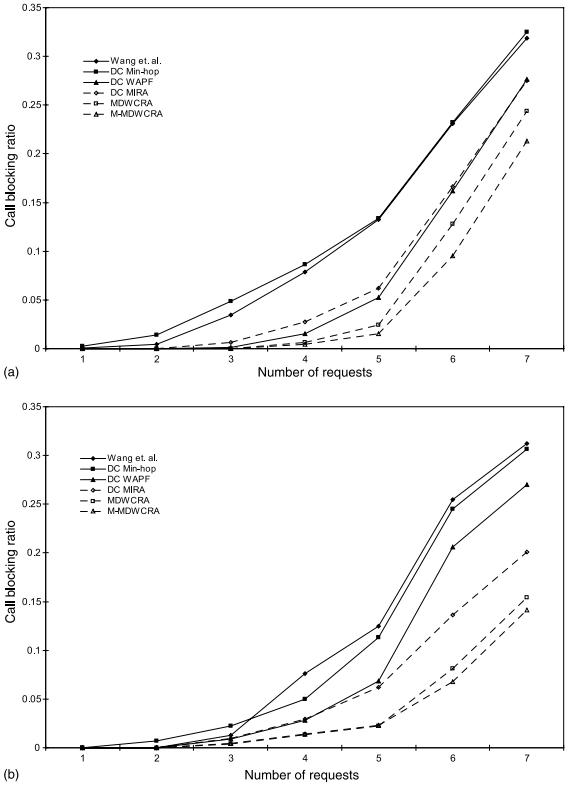


Fig. 8. Enlarged parts of Fig. 7: (a) evenly distributed traffic, (b) unevenly distributed traffic.

mentioned in Section 5, x is a positive integer which can take the value of $10d_{st}$, where d_{st} is the distance between s and t, for EDSP and EBF to be practical [5]. The order of $O(p)$ for a specific region is usually small, of the order of $O(x)$. Therefore $O(np \log n)$ is much smaller than $O(x^2n^2)$ and $O(xmn)$, which makes the total cost of MDWCRA just slightly higher than that of DC Min-Hop and DC WAPF. Although in the worst case, the number of paths searched by the M-MDWCRA algorithm is the same as the number of edges in the graph, in our simulations with different topologies and system parameters we find that the number is generally around two times the number of paths searched by the original MDWCRA algorithm. Table 1 summarizes our simulation results, comparing the number of critical links identified by MDWCRA and M-MDWCRA in different cases. It is important to note that when the ingress–egress pairs are not specified, all 56 links in MIRA topology are identified as critical by both algorithms. This is because the two nodes on either side of any link will become an ingress–egress pair at some point. It is the same with the expanded ANSNET topology.

In Section 5, we introduced three definitions of link weight for MDWCRA. After conducting simulation runs for various configurations, we found the performance under the three definitions is very close. For clarity of presentation, we selected MDWCRA under link weight definition 3 as the representative of the whole set. M-MDWCRA uses the same definition.

We use the *call blocking ratio* as the performance metric to compare the performance of different algorithms. Several experiments were conducted for each configuration, each time with a

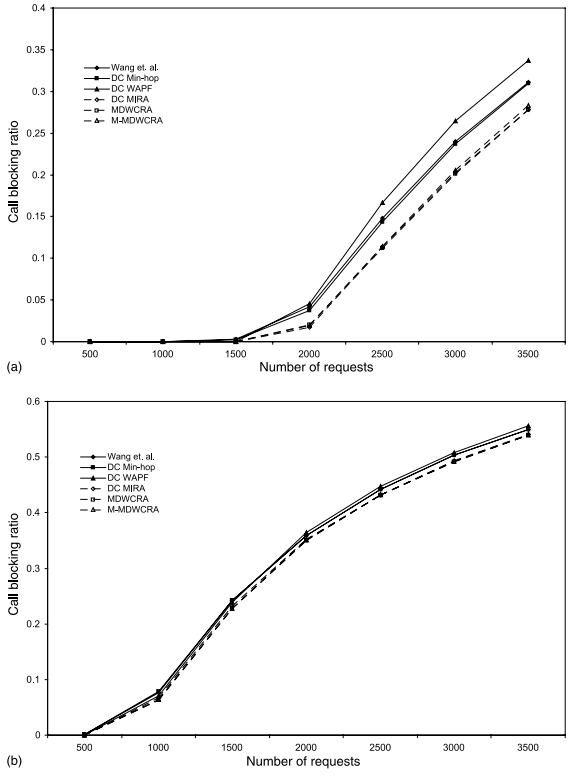


Fig. 9. Call blocking ratio as function of the number of requests. MIRA topology with non-specified ingress–egress nodes, $B \in [1–5]$, $D \in [25–35$ ms]: (a) evenly distributed traffic, (b) unevenly distributed traffic.

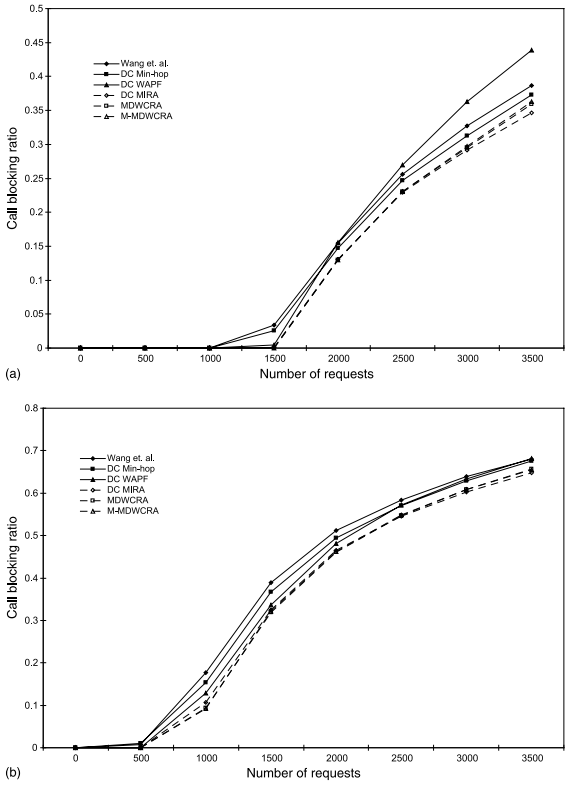


Fig. 10. Call blocking ratio as function of the number of requests. Expanded ANSNET topology with non-specified ingress–egress nodes, $B \in [1–5]$, $D \in [30–40$ ms]: (a) evenly distributed traffic, (b) unevenly distributed traffic.

different random seed. The results presented are the average of multiple runs.

Our first set of results is presented in Fig. 5(a) for evenly distributed traffic load and Fig. 5(b) for unevenly distributed traffic load for the MIRA topology with specified ingress–egress nodes. In these figures, the call blocking ratio is plotted as a function of the number of requests. For the even load, the requests were uniformly generated between the four source–destination pairs. For the uneven load, 80% of the requests were distributed between the two pairs (S_0, D_0) and (S_1, D_1). We examined the performance of the different algorithms for these configurations with bandwidth constraint $B \in [1–5]$ and delay constraint $D \in [25–35$ ms], [45–55 ms], [65–75 ms] respectively. We found that with increasing number of requests, the call blocking ratio increases consistently for all the algorithms. We also noticed that the performance ranking of these algorithms by the call blocking ratio remains the same independent of D. We have selected $D \in [25–35$ ms] as the representative value.

We see that MDWCRA and M-MDWCRA exhibit the best performance among all the algorithms. When the number of requests is small, the performance of all the algorithms is close because most of the requests can be accommodated when the load is light. This is also true when the number of requests is large since the network is saturated and a high percentage of the requests are blocked. We need also note that the small difference in call blocking ratios when the load is heavy actually corresponds to a significant difference in number of blocked requests. Therefore, the difference in call blocking ratio is most apparent under medium

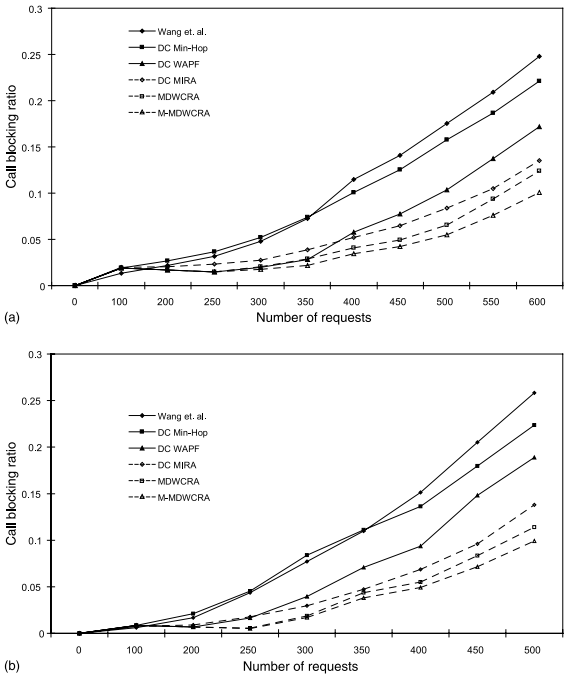


Fig. A.1. MIRA topology with specified ingress–egress nodes: (a) evenly distributed traffic, (b) unevenly distributed traffic.

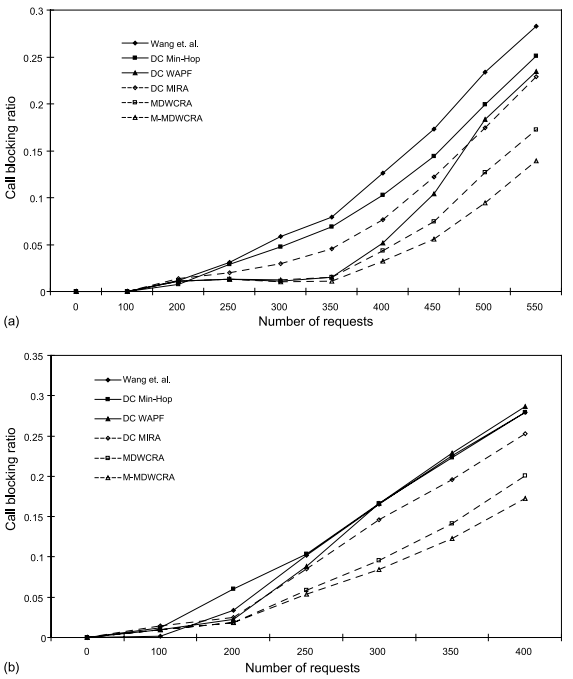


Fig. A.2. Expanded ANSNET topology with specified ingress–egress nodes: (a) evenly distributed traffic, (b) unevenly distributed traffic.

load. We magnified this region of Fig. 5(a) and (b) with blocking ratio below 30% and show them in Fig. 6(a) and (b) respectively to better illustrate the performance difference among different algorithms.

We see that in most cases the Wang and Crowcroft algorithm [22] performs the worst for both the even and uneven workload. This algorithm keeps using the least delay path of each *S–D* pair until it is used up and then tries to find an alternate path. The least delay path is obviously the best one to satisfy the delay requirement. However, this reduces the chance of accepting future requests with tight delay constraints. Furthermore, the links along the least delay path of one ingress–egress pair tend to have small link delays and thus have high probability of being on the least delay paths of other ingress–egress pairs. Therefore routing a request along the least delay path of one pair may also reduce the bandwidth of the least delay path of some other pairs.

The DC Min-Hop algorithm is the second worst in most cases among the algorithms studied. It finds an alternate path only when the shortest path of each *S–D* pair is used up. However it does not consider the information of ingress–egress nodes. The heavily loaded links along the shortest path may make it impossible to satisfy future requests between certain ingress–egress pairs. Since DC Min-Hop keeps using these links, it causes more future calls to be blocked than other algorithms.

The DC MIRA algorithm exhibits better performance than the above two algorithms. This is because DC MIRA utilizes knowledge of ingress–egress pairs to identify critical links. This works well especially when the traffic load is low. However, when the traffic load increases, the critical links in terms of bandwidth are not necessarily critical for delay. In fact, protecting them may decrease the chance of accepting future requests.

The performance of DC WAPF algorithm is surprisingly good. For the case of even load, DC WAPF performs much better than the above two algorithms and is almost as good as the MDWCRA algorithm. Since DC WAPF always chooses

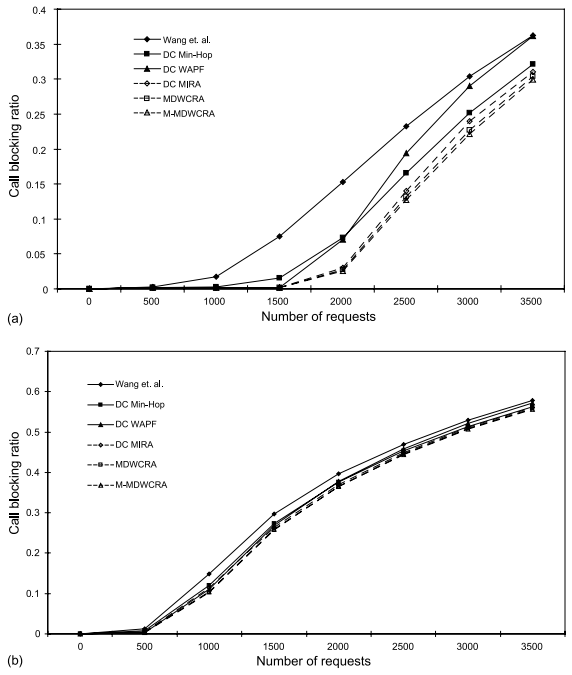


Fig. A.3. MIRA topology without specified ingress–egress nodes: (a) evenly distributed traffic, (b) unevenly distributed traffic.

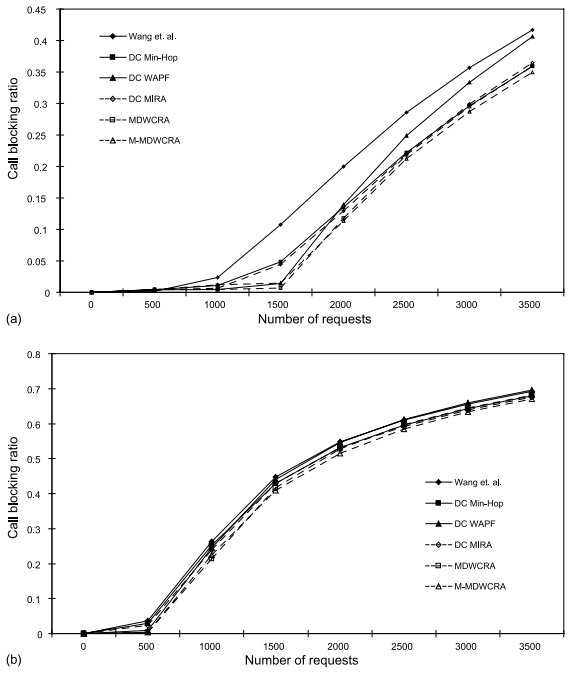


Fig. A.4. Expanded ANSNET topology without specified ingress–egress nodes: (a) evenly distributed traffic, (b) unevenly distributed traffic.

the widest available path, when the variation of link bandwidth in the network is small, the algorithm alternately chooses paths for each ingress–egress pair. With an evenly distributed load, the attempt of load-balancing the network traffic is effective as seen in Fig. 6(a). When the traffic load is unevenly distributed (Fig. 6(b)), the performance of DC WAPF is not as good compared to the case of even load. As shown in Fig. 6(b), DC WAPF blocks more requests than DC MIRA when the total request number is large (700 or above). This shows that balancing the traffic while the actual workload is uneven does not improve performance.

Both MDWCRA and M-MDWCRA perform better than the other algorithms for both even and uneven workload distributions. Both schemes attempt to first use links that are not critical to future requests for each ingress–egress pair. By deferring loading of the critical links, the potential of each ingress–egress pair to satisfy future requests is effectively preserved. Moreover, the weights of the links in the algorithms are assigned according to their criticality. The computation of least weight path maximizes the number of future requests accepted.

M-MDWCRA performs the best, improving MDWCRA by 20% with respect to call blocking ratio. By eliminating only the bottleneck link of the least delay path in each round more critical links will be protected. These critical links are assigned weights according to their criticality. Routing requests along the least weight paths makes use of links that are less critical. Therefore the probability of accommodating more future requests is improved.

Figs. 7 and 8 presents the call blocking ratio distribution of all the algorithms in the expanded ANSNET topology with specified ingress–egress nodes under the even load (Figs. 7(a) and 8(a)) and the uneven load (Figs. 7(b) and 8(b)). For the even load case, the requests are uniformly generated between the five source–destination pairs. For the uneven load case, 80% of the requests are distributed between the two pairs (S_0, D_0) and (S_3, D_3).

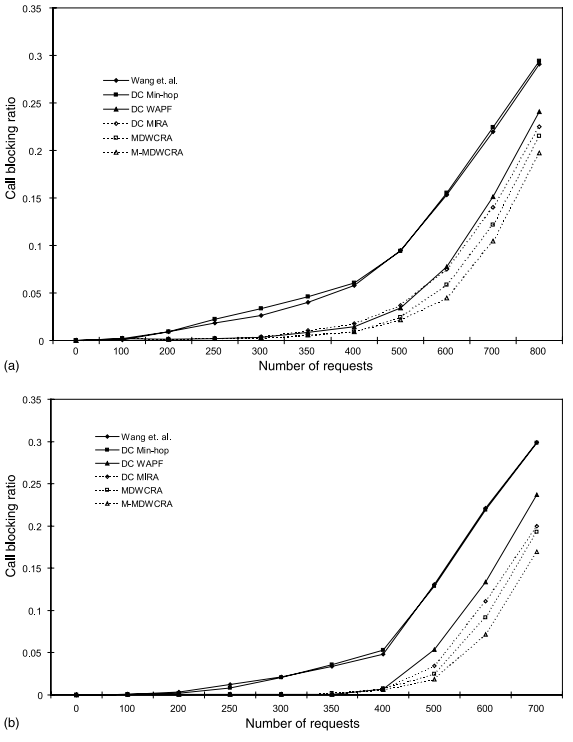


Fig. A.5. MIRA topology with specified ingress–egress nodes: (a) evenly distributed traffic, (b) unevenly distributed traffic.

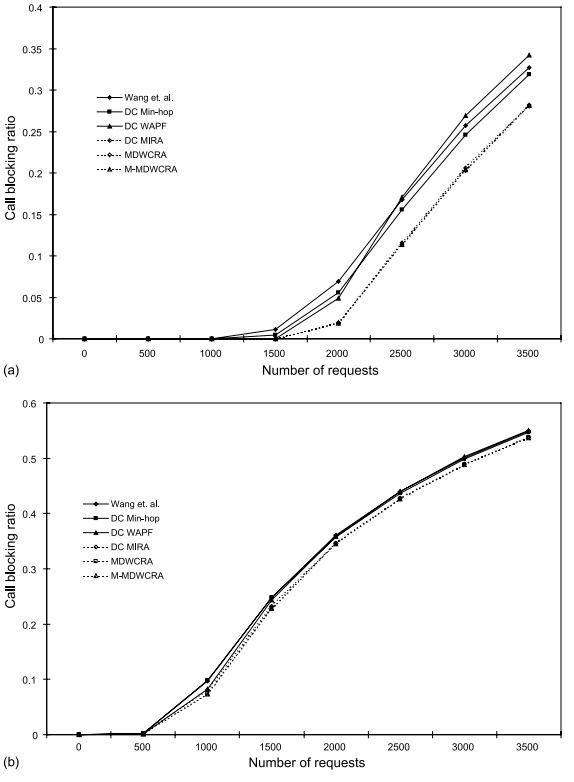


Fig. A.6. MIRA topology without specified ingress–egress nodes: (a) evenly distributed traffic, (b) unevenly distributed traffic.

The ranges of bandwidth and delay constraints are $B \in [1–5]$ and $D \in [30–40$ ms] respectively.

It is seen that the relative performance characteristics of the algorithms, except DC WAPF, are similar to those in the MIRA topology with specified ingress–egress nodes (see Figs. 6 and 8). In Fig. 8(a), both MDWCRA and M-MDWCRA exhibit much better performance than DC WAPF, while the curves of MDWCRA and DC WAPF are almost identical in Fig. 6(a). This is because the size of MIRA topology is small and there is no much delay variation. Therefore, the path selection of MDWCRA is mostly dominated by bandwidth and the widest path tends to be selected. This behavior is similar to DC WAPF. However, the expanded ANSNET topology is bigger and the delay variations among different paths are larger. In this situation, the advantage of MDWCRA in considering both delay and bandwidth is more clearly seen.

Figs. 9 and 10 summarize the call blocking ratio as a function of the number of requests for the MIRA topology and the expanded ANSNET topology without specified ingress–egress nodes. Figs. 9(a) and 10(a) show the case with evenly distributed workload where the source–destination pair of a request is randomly selected from all the nodes. Figs. 9(b) and 10(b) are for uneven load with 64% of the traffic distributed in four source–destination pairs. These four heavily loaded pairs are uniformly selected from all possible pairs formed by the nodes.

We notice that the performance of the algorithms in topologies with non-specified ingress–egress nodes is somewhat different from that in topologies with explicit ingress–egress nodes. In a topology where each node acts as a potential source or destination, each link in the network is crucial since the end points of that link form a potential source–destination pair. This means that each link has the property that whenever a path is

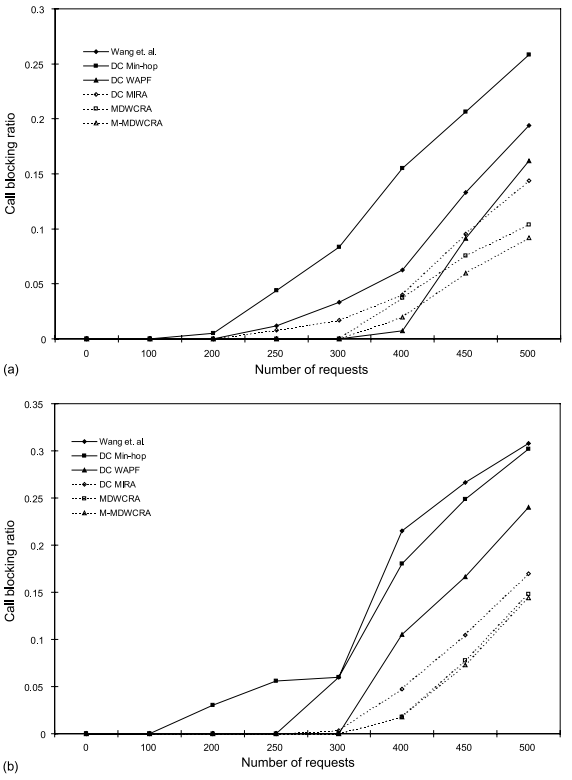


Fig. A.7. Expanded ANSNET topology with specified ingress–egress nodes: (a) evenly distributed traffic, (b) unevenly distributed traffic.

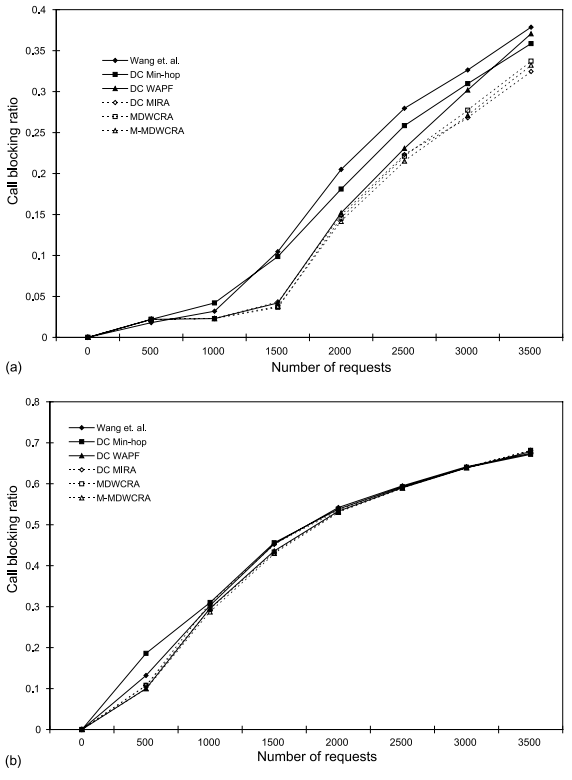


Fig. A.8. Expanded ANSNET topology without specified ingress–egress nodes: (a) evenly distributed traffic, (b) unevenly distributed traffic.

routed over that link, the number of requests that can be accepted between one or more ingress–egress pairs decreases.

For an evenly distributed workload shown in Figs. 9(a) and 10(a), we see that the Wang and Crowcroft algorithm performs the worst and the M-MDWCRA the best. When the load is light, the performances of all the algorithms (except the Wang and Crowcroft) are similar since most requests can be accommodated except those with very tight delay constraints. Under a medium load the behavior of the algorithms is similar to that in topologies with explicit ingress–egress. When the load gets heavy, the DC Min-Hop algorithm exhibits better performance than the other algorithms except MDWCRA and M-MDWCRA. The performance of MDWCRA and M-MDWCRA is always the best under all workload conditions in our experiments. When ingress–egress pairs are not specified, MDWCRA treats all the source–destination pairs formed by the nodes in the network equally. It performs very well since it is able to protect those links that would affect the most number of potential source–destination pairs. M-MDWCRA is better than MDWCRA in all workload ranges experimented because it protects more links which are likely to become critical in the future. However, the improvement in blocking ratio is not as significant compared to the case when ingress–egress pairs are specified. This can be explained by Table 1. When the ingress–egress pairs are not specified, all the links will eventually be identified as critical by MDWCRA and M-MDWCRA. Therefore, M-MDWCRA's advantage of protecting more critical links is lost.

For the uneven load presented in Figs. 9(b) and 10(b), we notice that the behaviors of the algorithms are similar to the case of an even load but

the differences between the algorithms become smaller. Although M-MDWCRA and MDWCRA are still the best, their performance is close to that of the other algorithms. This shows that the strategy of treating each possible source–destination pair equally while the actual traffic load is unbalanced does not effectively bring out the advantage of MDWCRA and M-MDWCRA.

8. Conclusions

In this paper we have presented a set of new algorithms for setting up bandwidth–delay constrained paths that exploit the knowledge of ingress–egress nodes. Routes are selected based on the notion of DWC so as to accommodate the maximum number of future requests. Simulation experiments have been conducted to examine the performance of the new algorithms using two different network topologies under evenly and unevenly distributed traffic load. We found that MDWCRA and M-MDWCRA perform the best compared to several existing algorithms. The difference in performance varies with the operating conditions. Future work will investigate the effects of topology and location of the ingress–egress pairs on performance.

Acknowledgement

This research work has been supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. AOE 98/99.EG01).

Appendix A

Part 1: Simulation results with link delays uniformly distributed in range [0, 50 ms]. $B \in [1–5]$, $D \in [100–115$ ms] for the MIRA topology and $B \in [1–5]$, $D \in [150–165$ ms] for the expanded ANSNET topology (Figs. A.1–A.4).

Part 2: Simulation results with link delays generated according to NSFNET T1 backbone measurements. $B \in [1–5]$, $D \in [40–55$ ms] for the MIRA topology and $B \in [1–5]$, $D \in [130–145$ ms] for the expanded ANSNET topology (Figs. A.5–A.8).

References

[1] G. Apostolopoulos, R. Guérin, S. Kamat, S.K. Tripathi, Server-based QoS routing, GLOBECOM'99, vol. 1b, 1999, pp. 762–768.
[2] D. Awduche, MPLS and traffic engineering in IP networks, IEEE Commun. 37 (12) (1999).
[3] Y. Bernet et al., A Framework for Differentiated Services, IETF Internet Draft, draft-ietf-diffserv-framework-02.txt, February 1999.
[4] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf, Computational Geometry, Algorithms and Applications, second ed., Springer, Berlin, 2000.
[5] S. Chen, Routing support for providing guaranteed end-to-end quality of service, Ph.D. thesis, Computer Science, University of Illinois at Urbana-Champaign, 1999.
[6] S. Chen, K. Nahrstedt, Distributed quality of service routing in high-speed networks based on selective probing, in: Proceedings of the 19th IEEE Tyrrbenian International Workshop on Digital Communications: Multimedia Communications, September 1998.
[7] S. Chen, K. Nahrstedt, An overview of quality of service routing for next-generation high-speed networks: problems and solutions, IEEE Network (November/December 1998).
[8] K.C. Clafly, G.C. Polyzos, H.W. Braun, Traffic characteristics of the T1 backbone of NSFNET, in: Proceedings of IEEE INFOCOM'93, San Francisco, CA, pp. 885–892, http://citeseer.nj.nec.com/claffy93traffic.html.
[9] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, L. Zhang, IDMaps: a global internet host distance estimation service, IEEE/ACM Trans. Network 9 (5) (2001) 525–540.
[10] A. Juttner, B. Szviatovszki, I. Mecs, Z. Rajko, Lagrange relaxation based method for the QoS routing problem, INFOCOM 2001, vol. 2, 2001.
[11] G. Feng, K. Makki, N. Pissinou, C. Douligeris, An efficient heuristic for delay-cost-constrained QoS routing, IEEE International Conference on Communications, 2001, ICC 2001, vol. 8, pp. 2603–2607.
[12] R. Guérin, A. Orda, D. Williams, QoS routing mechanisms and OSPF extensions, in: Proceedings of IEEE GLOBECOM'97, vol. 3, 1997, pp. 1903–1908.
[13] K. Kar, M. Kodialam, T.V. Lakshman, Minimum interference routing of bandwidth guaranteed tunnels with MPLS traffic engineering applications, IEEE J. Selected Areas Commun. 18 (12) (2000) 2566–2579.
[14] M. Kodialam, T.V. Lakshman, Minimum interference routing with applications to MPLS traffic engineering, IEEE INFOCOM 2000, March 2000.
[15] K. Lui, K. Nahrstedt, S. Chen, Hierarchical QoS routing in delay-bandwidth sensitive networks, in: Proceedings of IEEE LCN 2000, Tampa, FL, November 2000.

[16] P.P. Mishra, H. Saran, Capacity management and routing policies for Voice over IP traffic, IEEE Network 14 (2) (2000) 20–27.
[17] H. De Neve, P. Van Mieghem, TAMCRA: a tunable accuracy multiple constraints routing algorithms, Comp. Commun. 23 (7) (2000) 667–679.
[18] D.S. Reeves, H.F. Salama, A distributed algorithm for delay-constrained unicast routing, IEEE/ACM Trans. Network. 8 (2) (2000) 239–250.
[19] H.F. Salama, D.S. Reeves, Y. Viniotis, Evaluation of multicast routing algorithms for real-time communication on high-speed networks, IEEE J. Selected Areas Commun. 15 (13) (1997) 332–345.
[20] S. Suri, M. Waldvogel, P.R. Warkhede, Profile-based routing: a new framework for MPLS traffic engineering, in: Proceedings of Quality of Future Internet Services 2001, pp. 138–157.
[21] S. Suri, M. Waldvogel, D. Bauer, P.R. Warkhede, Profile-based routing and traffic engineering, Comp. Commun. 25 (2002).
[22] Z. Wang, J. Crowcroft, Quality of service routing for supporting multimedia applications, IEEE J. Selected Areas Commun. 14 (7) (1996) 1228–1234.
[23] Y. Yang, J.K. Muppala, S.T. Chanson, Quality of service routing algorithms for bandwidth–delay constrained applications, in: Proceedings of the 9th IEEE International Conference on Network Protocols (ICNP 2001), Riverside, CA, USA, 11–14 November 2001.

Yi Yang received her B.Sc. degree in Computer Science from Fudan University, Shanghai, China in 1999, and her M.Phil. degree in Computer Science from Hong Kong University of Science and Technology in 2001.

She is currently a Ph.D. student in Computer Science at the University of California, Los Angeles. Her research interests include wireless and mobile networks, Internet protocols and applications.

Lei Zhang is a Ph.D. candidate of the Department of Computer Science at the Hong Kong University of Science and Technology. His academic advisors are Prof. Samuel Chanson and Dr. Jogesh Muppala. He has received a B.S. in Computer Science from Tsinghua University, Beijing, P.R.China in 2000. His current research interests include Quality of Service routing, especially QoS routing with imprecise information, ad hoc networks, P2P networks, and VPN networks.

Jogesh Muppala received the Ph.D. degree in Electrical Engineering from Duke University, Durham, NC in 1991, the M.S. degree in Computer Engineering from The Center for Advanced Computer Studies, University of Southwestern Louisiana, Lafayette, LA in 1987 and the B.E. degree in Electronics and Communication Engineering from Osmania University, Hyderabad, India in 1985.

He is currently an associate professor in the Department of Computer Science, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. He also served as the Undergraduate Studies Director in the Computer Science Department for two years from August 1999 to August 2001.

He was previously a Member of the Technical Staff at Software Productivity Consortium (Herndon, Virginia, USA) from 1991 to 1992, where he was involved in the development of modeling techniques for systems and software. While at Duke University, he participated in the development of two modeling tools, the Stochastic Petri Net Package (SPNP) and the symbolic Hierarchical Automated Reliability and Performance Evaluator (SHARPE), both of which are being used in several universities and industry in the USA.

He was the program co-chair for the 1999 Pacific Rim International Symposium on Dependable Computing held in Hong Kong in December 1999. He has also served on program committees of many international conferences.

He was awarded the Teaching Excellence Appreciation Award by the Dean of Engineering, HKUST.

He is a Senior Member of IEEE, IEEE Computer Society and IEEE Communications Society.

Samuel Chanson received his B.Sc. degree in Electrical Engineering from Hong Kong University in 1969, and his M.Sc. and Ph.D. degrees in Electrical Engineering and Computer Sciences from the University of California, Berkeley in 1971 and 1975 respectively. He was a faculty member in the School of Electrical Engineering at Purdue University for two years. Then he moved to the Department of Computer Science at the University of British Columbia, where he became a full Professor and Director of the Distributed Systems Research Laboratory. In 1993 he joined the Computer Science Department at the Hong Kong University of Science and Technology. He is an Editor of IEEE/ACM Transactions on Networking, the Journal of Computing and Information, and also New Generation Computing. He also serves on the IFIP TC6/WG6.1 Committee. He has chaired many international conferences on communication protocols and distributed systems and has published more than 150 papers in these areas. In Hong Kong, He is Chairman of Information Security and Forensics Society (ISFS), Chairman of the Internet Business Consortium (IBC), and Director of the Cyberspace Center. He has been widely consulted by industry and government institutes on Internet security, e-commerce, and communication technologies both in North America and in Asia.