Quality of Service Routing Algorithms for Bandwidth-Delay Constrained Applications

Yi Yang, Jogesh K. Muppala and Samuel T. Chanson
Department of Computer Science
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong

Abstract

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.

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) [11]. 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.

Most existing bandwidth-delay-constrained routing algorithms [6, 10, 14] 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 in [9] to reduce the number of rejected connection setup requests. 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 or Bellman-Ford 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 delay-weighted capacity routing algorithms (MDWCRA), 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,

1092-1658/01 $17.00 © 2001 IEEE

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, 8, 9, 11]. Two commonly used algorithms are the minimum hop (Min-Hop) [8] and the widest path available path first (WAPF) [11]. 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 et. al. algorithm [14] the links with available bandwidth less than the bandwidth requirement 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) [14] finds a feasible widest available path between the ingress and egress nodes. If multiple paths exist, the path with the minimum delay, called the shortest widest path, is selected.

The Minimum Interference Routing algorithm (MIRA) [9] 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. *Critical* links are identified as those links for which 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.

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, 12, 13]. 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:

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

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 $LP_{st} = \{LP_{st}^1 ... LP_{st}^i ... LP_{st}^{k_{st}}\}$. LP_{st}^i is the least delay path computed using the graph where the links belonging to LP_{st}^1, ..., 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}.

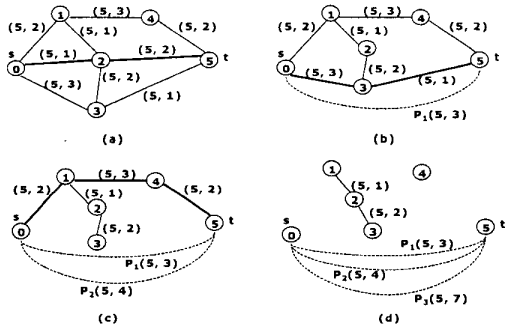


Figure 1: Illustrative example. Vector associated with each link is (bandwidth, delay).

An example is shown in Fig. 1. In this figure, the first least delay path between s ($s = 0$) and t ($t = 5$) is computed, which is $P_1(0, 2, 5)$ with bandwidth 5 and delay 3. Path P_1 is abstracted as a dotted link from s to t shown in Fig. 1(b). The second least delay path $P_2(0, 3, 5)$ 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(d), 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 capacity. We define the *delay-weighted capacity* (DWC) for each pair (s, t) below.

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

$$DWC_{st} = \sum_{LP_{st}^i \in LP_{st}} \frac{B_{st}^i}{D_{st}^i}$$

An *optimal path* for a request can be computed as the route that maximizes a weighted sum of the DWC value between every ingress-egress pair.

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 $C_{st} = \{C_{st}^1, ..., C_{st}^i, ..., C_{st}^{k_{st}}\}$, where C_{st}^i consists of all the bottleneck links for the least delay path LP_{st}^i.

It is clear we should avoid routing paths on critical links as much as possible. We do this by assigning weights to critical links that are an increasing function of their *criticality*. An extended Dijkstra or Bellman-Ford algorithm is used to compute the least weight path.

5. Routing Algorithm

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_{st}^i at each round in the iteration, which takes $O(n \log n)$. Critical links for LP_{st}^i can be found in $O(n)$ since LP_{st}^i 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 at 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 extended Dijkstra (EDSP) or Bellman-Ford (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. The routing algorithm is detailed below, where α_{st}^i is a property associated with least delay path LP_{st}^i. We shall discuss how to set the value of α_{st}^i later.

Maximum Delay-Weighted Capacity Routing Algorithm (MDWCRA)

1. Compute the delay-weighted capacity (DWC) values for all $(s,t) \in P$.
2. Compute the set of critical links C_{st} for all $(s,t) \in P$.
3. Compute the link weights $w_l = \sum_{(s,t):l \in C_{st}^i} \alpha_{st}^i$ $\forall l \in E$.
4. Eliminate all links that have residual bandwidth less than B and form a reduced network.
5. 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.
6. Route the request from a to b along this delay-constrained least-weight path and update the residual capacities of the network.

By varying the value of α_{st}^i, different definitions of link weight can be obtained as follows:

1. $w_l = \sum_{(s,t):l \in C_{st}^i} 1$, with $\alpha_{st}^i = 1$.

The weight of link l represents the number of ingress-egress pairs for which link l is critical.

2. $w_l = \sum_{(s,t):l \in C_{st}^i} \frac{1}{D_{st}^i}$, with $\alpha_{st}^i = \frac{1}{D_{st}^i}$.

The link weight is inversely proportional to the end-to-end delay value of the least delay path.

3. $w_l = \sum_{(s,t):l \in C_{st}^i} \frac{1}{B_{st}^i \times D_{st}^i}$, with $\alpha_{st}^i = \frac{1}{B_{st}^i \times D_{st}^i}$.

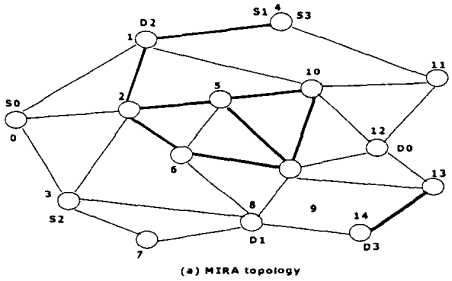
The link weight is inversely proportional to the product of the bandwidth and the end-to-end delay of the least delay path.

Now we analyze the time complexity of MDWCRA. As discussed above, Step 1 and 2 take $O(np \log n)$. Step 3 can be piggybacked on Step 2. So no additional complexity is introduced. Step 4 costs $O(m)$. Step 5 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.

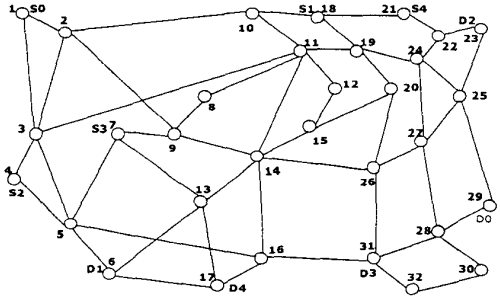
6. Network Configuration

Two different network topologies were used to compare the different routing algorithms. The first topology, adopted from [9], is called the MIRA topology and consists of 15 nodes as shown in Fig. 2(a). All the links are considered bidirectional. There are two different kinds of links in the network: The light links have a capacity of 12 units and the dark links have a capacity of 48 units. The delay values of the links are uniformly distributed in the range of [0...50ms]. A subset of the nodes in the network acts as the ingress-egress pairs. In the MIRA topology, four such pairs are considered.

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. 2(b). Each link has the capacity of 12 units. The delay values of the links are uniformly distributed in the range of [0...50ms]. Five pairs in this topology are considered as ingress-egress pairs.



(a) MIRA topology



(b) Expanded ANSNET topology

Figure 2: Network Topologies

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 heavy-loaded pairs are explicitly specified in the MIRA and expanded ANSNET topology with specified ingress-egress nodes. In the case without specified ingress-egress nodes, the heavy-loaded pairs are randomly selected from all possible pairs.

The bandwidth requirement of a request is uniformly distributed between 1 and 5 units. The delay requirement is generated from different ranges, viz., [50...65ms], [75...90ms], [100...115ms], [125...140ms] and [150...165ms]. 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:

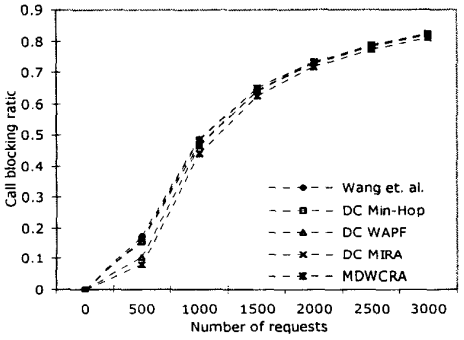
1. The Wang et. al. algorithm with complexity of $O(n\log n + m)$ [14]
2. The delay-constrained Min-Hop (DC Min-Hop) algorithm with complexity of $O(x^2n^2)$ or $O(xmn)$
3. The delay-constrained WAPF (DC WAPF) algorithm with complexity of $O(x^2n^2)$ or $O(xmn)$
4. The delay-constrained MIRA (DC MIRA) algorithm with complexity of $O(pn^2\sqrt{m} + x^2n^2)$ or $O(pn^2\sqrt{m} + xmn)$
5. MDWCRA with complexity of $O(np\log n + x^2n^2)$ or $O(np\log n + xmn)$

The original Min-Hop [8], WAPF [11] and MIRA [9] algorithms deal with the case of setting up bandwidth-guaranteed connections between given source and destination pairs. These three algorithms utilize the well known Dijkstra or Bellman-Ford algorithm to calculate the feasible path. We modify them by using the extended Dijstra (EDSP) or Bellman-Ford (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 mentioned in Section 5, x is a positive integer which can take the value of

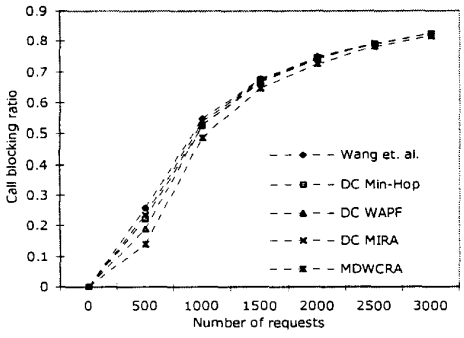
$10d_{st}$, where d_{st} is the distance between s and t, for EDSP and EBF to be practical [5]. The order of $O(p)$ for a specific region is usually small, of the order of $O(x)$. Therefore $O(np\log n)$ is much smaller than $O(x^2n^2)$ and $O(xmn)$, which makes the total cost of MDWCRA just slightly higher than that of DC Min-Hop and DC WAPF.

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.

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



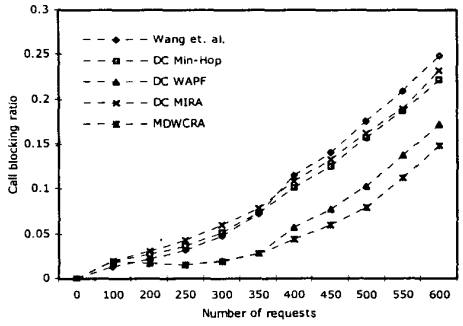
(a) Evenly distributed traffic



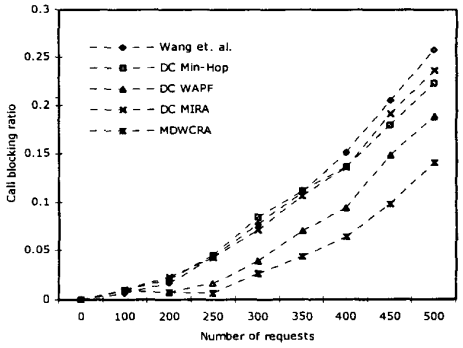
(b) Unevenly distributed traffic

Figure 3: Call blocking ratio as function of the number of requests. MIRA topology with specified ingress-egress nodes, $B\in$ [1...5], $D\in$ [100...115ms]

Our first set of results is presented in Fig. 3(a) for evenly distributed traffic load and Fig. 3(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 $D\in$ [50...65ms], [75...90ms], [100...115ms], [125...140ms], [150...165ms] 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 $D\in$ [100...115ms] as the representative value.



(a) Evenly distributed traffic



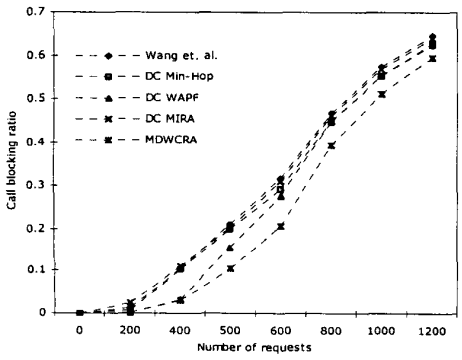
(b) Unevenly distributed traffic

Figure 4: Call blocking ratio as function of the number of requests. MIRA topology with specified ingress-egress nodes, $B\in$ [1...5], $D\in$ [100...115ms]

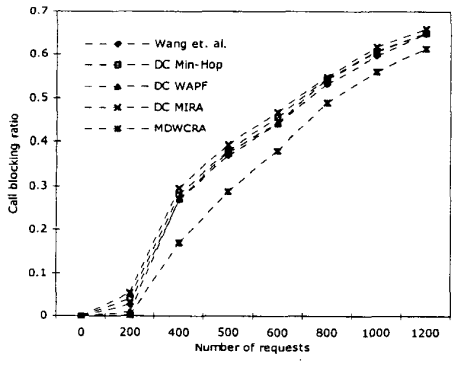
We see that MDWCRA exhibits the best performance among all the algorithms. When the number of requests is small, the performance of all the algorithms is close because most of the requests can be accommodated when the load is light. This is also true when the number of requests is large since the network is saturated and a high percentage of the requests are blocked. The difference in call blocking ratio is most apparent under medium load. We

magnified this region of Figs. 3(a) and 3(b) with blocking ratio below 30% and show them in Figs. 4(a) and 4(b) respectively to better illustrate the performance differences among the different algorithms. For network service providers and users, a network with low blocking ratio is of the most interest. In all cases of these figures, the maximum blocking ratio is no more than 5% worse than the average value.

We see that in most cases the Wang et. al. algorithm 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.



(a) Evenly distributed traffic

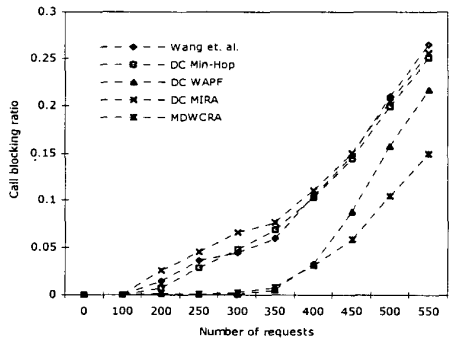


(b) Unevenly distributed traffic

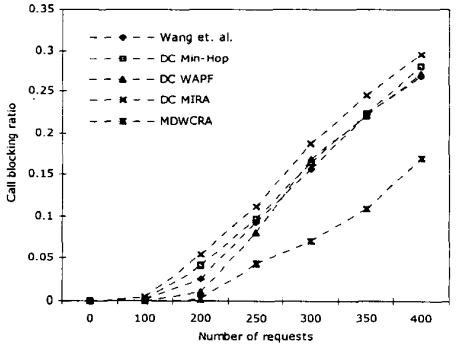
Figure 5: Call blocking ratio as function of the number of requests. Expanded ANSNET topology with specified ingress-egress nodes, $B\in$ [1...5], $D\in$ [150...165ms]

The DC MIRA algorithm performs very close to the Wang et. al. algorithm. Although DC MIRA utilizes knowledge of ingress-egress pairs to identify critical links for each pair, the critical links are defined considering only the bandwidth. However, the critical links in terms of bandwidth are not necessarily critical for delay. In fact, protecting them may decrease the chance of future requests being accepted.

The DC Min-Hop algorithm is the third 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.



(a) Evenly distributed traffic



(b) Unevenly distributed traffic

Figure 6: Call blocking ratio as function of the number of requests. Expanded ANSNET topology with specified ingress-egress nodes, $B\in$ [1...5], $D\in$ [150...165ms]

The performance of DC WAPF algorithm is in between the algorithms. For the case of even load, DC WAPF performs much better than the above three algorithms. Since DC WAPF always chooses 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. 4(a). When the traffic load is unevenly distributed (Fig. 4(b)), the performance of DC WAPF is not as good compared to the case of even load. This shows that balancing the traffic while the actual workload is uneven does not improve performance.

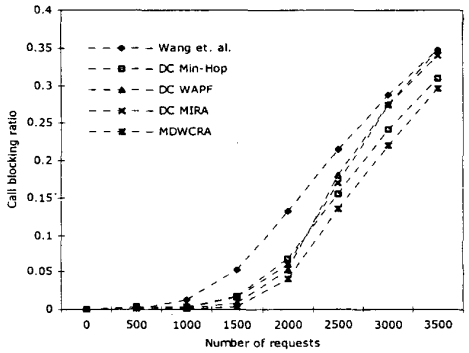
MDWCRA performs the best for both distributions of workload. This algorithm attempts to first use links that are not critical to future requests for each ingress-egress pair. By deferring loading of the critical links, the potential of each ingress-egress pair to satisfy future requests is effectively 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.

Figs. 5 and 6 present the call blocking ratio distribution of all the algorithms in the expanded ANSNET topology with specified ingress-egress nodes under the even load (Figs. 5(a) and 6(a)) and the uneven load (Figs. 5(b) and 6(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). The ranges of bandwidth and delay constraints are $B \in [1...5]$ and $D \in [150...165\text{ms}]$ respectively. It is seen that the performance characteristics of all algorithms are similar to those in the MIRA topology with specified ingress-egress nodes.

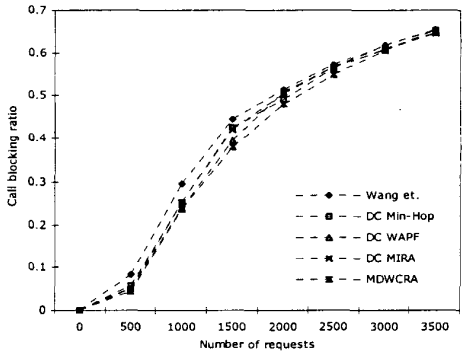
Figs. 7 and 8 summarize the call blocking ratio as a function of the number of requests for the MIRA topology and the expanded ANSNET topology without specified ingress-egress nodes. Figs. 7(a) and 8(a) show the case with evenly distributed workload where the source-destination pair of a request is randomly selected from all the nodes. Figs. 7(b) and 8(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 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 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. 7(a) and 8(a), we see that the Wang et. al. algorithm performs the worst and the MDWCRA the best. When the load is light, the performances of all the algorithms (except the Wang et. al.) are similar since most requests can be accommodated except those with very tight delay constraints. Under a medium load the behaviors of the algorithms are similar with those 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. The performance of MDWCRA is always the best in all the workload ranges in our experiments. When possible 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.



(a) Evenly distributed traffic

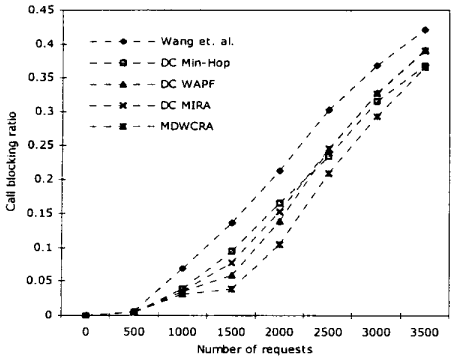


(b) Unevenly distributed traffic

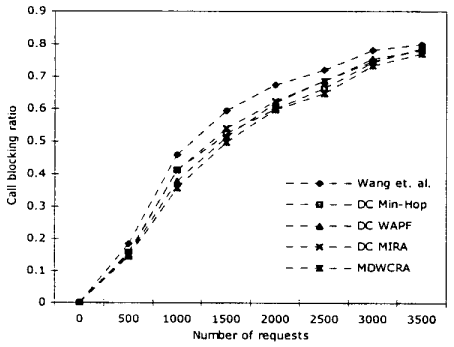
Figure 7: Call blocking ratio as function of the number of requests. MIRA topology with non-specified ingress-egress nodes, $B \in [1...5]$, $D \in [100...115\text{ms}]$

For the uneven load presented in Figs. 7(b) and 8(b), we notice that the behavior of the algorithms are much like the case with an even load but the differences between the algorithms become smaller.

Although MDWCRA is still the best, its performance is close to the others. 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.



(a) Evenly distributed traffic



(b) Unevenly distributed traffic

Figure 8: Call blocking ratio as function of the number of requests. Expanded ANSNET topology with non-specified ingress-egress nodes, $B \in [1...5]$, $D \in [125...140ms]$

8. Conclusions

In this paper we presented a set of new algorithms for setting up bandwidth-delay constrained paths that exploit the knowledge of ingress-egress nodes. They route paths based on the notion of delay-weighted capacity so as to accommodate the maximum number of future requests. Simulation experiments were conducted to examine the performance of the new algorithms using two different network topologies under evenly and unevenly distributed traffic load. We found that the MDWCRA scheme performs the best compared to several other 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.

Acknowledgements

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).

References

[1] G. Apostolopoulos, R. Guérin, S. Kamat, S. K. Tripathi, Server Based QoS Routing, *GLOBECOM'99*, vol. 1b, pp. 762-8, 1999.

[2] D. Awduche, MPLS and Traffic Engineering in IP Networks, *IEEE Communications*, vol. 37, no. 12, Dec. 1999.

[3] Y. Bernet, et, A Framework for Differentiated Services, *IETF Internet Draft, draft-ietf-diffserv-framework-02.txt*, Feb. 1999.

[4] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf, *Computational Geometry, Algorithms and Applications, second edition*, Springer, 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, *Proc. 19th IEEE Tyrrbenian Int. Wkshp. on Digital Communications: Multimedia Communications*, Sep. 1998.

[7] S. Chen, K. Nahrstedt, An Overview of Quality of Service Routing for Next-Generation High-Speed Networks: Problems and Solutions, *IEEE Network*, Nov./Dec. 1998.

[8] R. Guérin, A. Orda, D. Williams, QoS Routing Mechanisms and OSPF Extensions, in *Proc. IEEE GLOBECOM'97*, vol. 3, pp. 1903-8, 1997.

[9] M. Kodialam, T. V. Lakshman, Minimum Interference Routing with Applications to MPLS Traffic Engineering, *IEEE INFOCOM 2000*, March 2000.

[10] K. Lui, K. Nahrstedt, S. Chen, Hierarchical QoS Routing in Delay-Bandwidth Sensitive Networks, in *Proc. IEEE LCN 2000*, Tampa, FL, November 2000.

[11] P. P. Mishra, H. Saran, Capacity Management and Routing Policies for Voice over IP Traffic, *IEEE Network*, vol. 14, no. 2, pp. 20-27, March/April 2000.

[12] H. De Neve, P. Van Mieghem, TAMCRA: a Tunable Accuracy Multiple Constraints Routing Algorithms, *Computer Communications*, vol. 23, no. 7, pp. 667-79, 15 Mar. 2000.

[13] D. S. Reeves, H. F. Salama, A Distributed Algorithm for Delay-Constrained Unicast Routing, *IEEE/ACM Trans. on Networking*, vol. 8, no. 2, pp. 239-50, Apr. 2000.

[14] Z. Wang, Crowcroft J. Quality-of-service Routing for Supporting Multimedia Applications, *IEEE J. on Selected Areas in Comm.*, vol. 14, no. 7, pp. 1228-34, Sep.1996.