Journal of Network and Systems Management, Vol. 10, No. 3, September 2002 (© 2002)

DORA: Efficient Routing for MPLS Traffic Engineering

R. Boutaba,[1,2] W. Szeto,[1] and Y. Iraqi[1]

This paper introduces DORA, a dynamic online routing algorithm for construction of bandwidth guaranteed paths in MPLS-enabled networks. The main objective of DORA is to place paths with reserved bandwidth evenly across the network in order to allow more future paths to be accepted into the network and to balance the traffic load. During path computation, the key operation in DORA is to avoid routing over links that (1) have high potential to be part of any other path, and (2) have low residual bandwidth available. Our simulation results based on unsuccessful path-setup ratio and successful path-reroutes upon link failure, show that DORA offers better performance than some sophisticated algorithms, while at the same time being less computationally expensive.

KEY WORDS: MPLS; constraint-based routing; traffic engineering.

1. INTRODUCTION

The exponential growth of the Internet has placed heavy burdens on network management and control operations. Adding more resources to the network may temporarily relieve congestion conditions, but it is not a cost-effective solution in solving resource contention problems in the long run. Furthermore, the trend toward providing differentiated classes of service adds another level of complexity to network management and operations. What service providers need are mechanisms to coordinate, control, and efficiently utilize existing resources to satisfy customer demand. Multi-protocol label switching (MPLS) and constraint-based routing (CR) are two key elements of the new Internet architecture and can be used to alleviate resource contentions and improve overall network utilization.

[1]Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1.
[2]To whom correspondence should be addressed at Department of Computer Science, University of Waterloo, 200 University Avenue West, Waterloo, Ontario, Canada N2L 3G1. E-mail: *rboutaba@bbcr.uwaterloo.ca*

1064-7570/02/0900-0309/0

One particular problem that providers are facing today is the on-demand setup of network tunnels with reserved bandwidths in backbone or transport networks. We address this issue by proposing a new routing algorithm called dynamic online routing algorithm (DORA). DORA aims to efficiently utilize existing network resources and minimize network congestions by carefully mapping paths across the network. The problem of establishing bandwidth guaranteed paths has been studied elsewhere [1–5]. Our work is inspired by the minimum interference routing algorithm (MIRA) [1], but performs better in terms of path-setup rejection ratio and rerouting percentage upon link/node failure with much less computation complexity.

The rest of this paper is structured as follows: Section 2 provides the necessary background information on MPLS and CR. Section 3 identifies some important design issues for routing algorithms in an MPLS network. In Section 4 we discuss related works. Section 5 describes the details of the DORA algorithm. Section 6 evaluates and compares DORA performance against other existing routing algorithms through simulations. Finally, Section 7 summarizes the important points in the paper and proposes future works.

2. BACKGROUND INFORMATION

In MPLS networks, when a packet enters the network, it is assigned a label at the ingress router and then forwarded along a preestablished path called the label switched path (LSP). Each LSP carries an aggregate of traffic flows, called a traffic trunk, that is generated by assigning incoming packets to different forwarding equivalent classes (FECs) at the ingress router. The packet classification decision is based on examining packet header information such as the source address field, the destination address field, and/or type-of-service field. All packets belonging to the same FEC are treated in an equivalent manner at each hop. Different forwarding granularities of FECs could be defined. An example of a fine granularity classification is to group packets with equivalent source IP address, source port number, destination IP address, and destination port number into the same FEC. An example of a coarse granularity classification is to group packets coming from the same input port of an ingress router into the same FEC. MPLS, together with CR, facilitates traffic-engineering operations. The goal of CR is to compute an explicit network path that meets a set of user requirements or constraints and optimizes some scalar metric such as the number of hops. The explicit route starts at an ingress/source router and ends at an egress/destination router. The product of a successful CR operation is a constraint-based routed LSP (CR-LSP), which is simply an LSP that is subjected to a set of constraints, such as bandwidth, jitter, or delay requirements.

3. DESIGN ISSUES OF CONSTRAINT-BASED ROUTING ALGORITHMS

In this section, we identify and discuss important issues pertaining to path selection algorithms within MPLS domains.

1. *Routing Constraints*: Constraints may include delay, jitter, loss ratio, bandwidth, administrative constraints, and so on. Computing an optimized path according to two or more additive (e.g. delay, jitter) and/or multiplicative constraints (e.g. loss ratio) has been proven to be NP-complete [6]. Furthermore, it is generally difficult to obtain accurate values for certain metrics such as delay and jitter. The general practice is to convert additive and multiplicative constraints into a bandwidth requirement. In the rest of this paper, we will focus on routing algorithms that compute paths subjected to bandwidth requirement.
2. *Online/Offline Routing*: CR algorithms are categorized as either online or offline. Offline CR requires a demand matrix as input and is usually performed in a separate server outside the network. The demand matrix describes precisely the amount of data to be transmitted for each source–destination pair of routers. The main objective of offline CR is to optimize network resource usage using a priori knowledge of traffic demand. Assuming the existence of a demand matrix is justifiable for network design and planning purposes; however it is difficult, if not impossible, to obtain an accurate demand matrix in the case of dynamic MPLS paths setup. Online CR, in turn, does not require a priori knowledge of the size and arrival time of each individual path setup request. This paper focuses on online CR, as it is the appropriate approach to solving the dynamic path setup problem in MPLS networks.
3. *Computational Complexity*: Online routing algorithms compute paths as setup requests arrive at the network in real time. Each incoming request is processed at the ingress router that typically operates at very high load. The time it takes for a path to be established in the network should be short. Therefore it is important for path computations to be as fast and as efficient as possible.
4. *Re-routing Performance*: Network topology changes are typically triggered by link or node failures. Re-routing can induce service interruption or delay for the end users. Also, since each path is associated with reserved bandwidth, rerouting may fail due to insufficient resources. Thus, an important performance metric of a routing algorithm is its ability to map traffic trunks onto the network in such a way that the number of paths affected by link or node failures is minimized.
5. *Link State Distribution*: In order to compute paths with specific bandwidth requirements, it is necessary for the node(s) that executes the routing

algorithm to have correct topology and resource information. There have been proposals to extend the current interior gateway protocols (IGPs) to carry additional information such as the capacity and the residual bandwidth of a link [7, 8]. In this paper, we assume that the necessary bandwidth-related information for routing purposes is propagated across the network via link state advertisements.

4. RELATED WORKS

The most widely used routing algorithm within MPLS domains is the shortest-path first (SPF) algorithm based on an administrative metric (usually the number of hops). In SPF, the shortest path between the source and the destination node is chosen. SPF is simple, yet it tends to send traffic flows onto the same set of nodes and links because it routes packets onto the same set of shortest paths until resources along those paths are saturated. This results in congestions in some parts of the network and unbalanced network resource utilization.

A more advanced routing algorithm is (MIRA) [1]. The key idea behind MIRA is to avoid routing over links that may "interfere" with potential future paths setup. These links are referred to as critical links, and they have the property that when their capacity is reduced by 1 bandwidth-unit, the maximum data flow between a source–destination node pair is also reduced by 1 bandwidth-unit. Hence the goal of MIRA is to find a feasible path that contains the least number of critical links. We believe that the idea of routing over noninterfering links is a sound and logical one; however MIRA has two weaknesses: computation complexity and unbalanced network utilization. First, suppose there are p source–destination pairs in a network with N nodes and M links. MIRA requires p maximum flow calculations to determine the set of critical links each time a path setup request arrives. In the worst case, every node is a source node for every other node, and so p becomes N^2. Since each maximum flow calculation is $O(N^3)$ [3], therefore the worst case runtime of MIRA is $O(N^5)$, which is several order of magnitudes higher than that of SPF (i.e. $O(MN)$). Second, suppose there are two distinct routes with the same residual bandwidth that connects the same source–destination pair. When a path setup request arrives, given sufficient resources, one of the two routes will be chosen to service this request. Afterwards, all the links in the other route become critical links according to the definition of critical links above. This implies that the same route will serve subsequent requests until saturation while the other route remains free. Therefore, given several distinct routes with enough residual bandwidth, MIRA may converge traffic flows onto a single route causing unbalanced network utilization because it does not take into account the current traffic load in routing decisions [1].

Other related works include profile-based routing [2], variations of OSPF routing schemes [3], explicit routing algorithms [4], and static routing algorithm

[5]. In profile-based routing, a "traffic profile" is used to solve a multicommodity network flow problem. A traffic profile contains an estimate of future demand for each source–destination pair in the network. It preallocates resources for each source–destination pair based on the solution obtained by solving the multicommodity problem. A new path will be established in the network only if there are sufficient reserved resources available for the associated commodity. In Ref. 3, authors proposed several ways to improve the routing performance of OSPF. The most significant result is a local search heuristic that computes the weight of each link according to an estimated demand matrix. The work on explicit routing algorithms is based on modeling the traffic engineering problem as an optimization problem with the objective of minimizing congestion and maximizing the potential for traffic growth [4]. The goal of the optimization problem is accomplished by minimizing the maximum utilization of each network link. Static routing algorithm attempts to balance network traffic distribution by keeping the utilization of each link below a predefined threshold a in an offline fashion [5]. The value of a is set administratively or based on past experience. The algorithm executes by remapping each LSP onto the network, starting with the highest priority LSP, until all LSPs have been considered. If there are remaining unmapped LSPs, increase the threshold a by a small amount, and repeat the LSP remapping procedure for unmapped LSPs only. If there are still links unmapped after the first two rounds, the threshold a will be increased again until all unmapped links are remapped or a reaches its maximum value of 1.

Both profile-based routing and local search heuristic necessitate a network traffic profile or demand matrix as an estimation of future demands. Explicit routing algorithms require the average traffic between each edge node as inputs for solving the associated mathematical formulations. Static routing aims to balance network traffic offline where minimizing path computation time is not an issue. These routing algorithms do not satisfy the design issues stated in Section 3, and therefore they will not be used in our performance evaluation section. Instead, we will focus on comparing SPF and MIRA against our proposed algorithm.

5. DYNAMIC ONLINE ROUTING ALGORITHM (DORA)

This section provides a detailed description of DORA and is organized as follows: First, we will point out a few important observations and ideas behind the design of DORA. Next, we will state the pseudo-code version of DORA.

In MPLS networks, an ingress (or source) node is an edge router that acts as an entrance point to the network, while an egress (or destination) node is an edge router that acts as an exit point from the network. We consider the problem of setting up bandwidth guaranteed MPLS tunnels between a known set of source and destination nodes. Each request arrives at the network to demand a path with reserved bandwidth (size of a request) to be set up between a source node and a

destination node. We assume that requests arrive one by one and there is no a priori information on the arrival time or the size of each request.

The operation of DORA is divided into two stages. The first stage calculates the path potential value (PPV) array associated with a source–destination pair and the second stage combines PPV with residual link bandwidth to form a weight value for each link, which is then used to compute a weight-optimized network path. The design of the first stage is based on the followign observation: There are many paths that data could take to flow between a given source and destination nodes, and some links are more likely to be included in these paths than are others. The potential of a link being more likely to be included in a path than other links is characterized by an associated PPV. More formally, for each source–destination pair, we associate each link with an integer called the PPV with an initial value zero. Each source–destination pair (S, D) is associated with an array, $PPV_{(S,D)}$. When a path could be constructed over a link L for a given source–destination pair (S1, D1), we reduce $PPV_{(S1,D1)}$(L) by 1. When a path could be constructed over the same link L for a different source–destination pair (S2, D2), we increment $PPV_{(S1,D1)}$(L) by 1. Since there are many paths that exist between a given source–destination pair, we consider only disjoint paths. The calculation of PPV arrays for each source–destination pair forms the first stage of the algorithm.

In the second stage of DORA, we will remove all links with a residual bandwidth less than the required bandwidth. The PPV and current residual bandwidth of each link are combined together to form the link weight. The content of the link weight is controlled by a parameter called BWP (for bandwidth proportion), which takes on real values between 0.0 and 1.0. For example, if BWP equals 0.7, this implies that 70% of the link weight is contributed by the link's residual bandwidth and the other 30% is contributed by the associated PPV. Finally, we run Dijsktra's algorithm to compute a weight-optimized path from the source node to the destination node on the residual topology.

5.1. DORA Pseudo-Code

Stage 1

1. For each source–destination pair (S, D), determine the set of all disjointed paths $DP_{(S,D)}$. One possible way is to use Dijsktra's algorithm to find a shortest path (in terms of number of hops) for (S, D), add this path to $DP_{(S,D)}$, and then remove all links that are part of the resulting path, and repeat these steps until D is no longer reachable from S.
2. For each source–destination pair (S, D), construct the $PPV_{(S,D)}$ array, and initialize all entries to zero. The size of the array is equal to the number of network links.

3. For the source–destination pair (S1, D1),
   (a) For each link L in the network, if L is part of any path in $DP_{(S1,D1)}$, subtract 1 from $PPV_{(S1,D1)}(L)$.
   (b) For all the source–destination pairs other than (S1, D1), inspect each link L and determine the number of times n that L appears in $DP_{(S,D)}$ where (S, D) is not equal to (S1, D1). Increment $PPV_{(S1,D1)}(L)$ by n.
4. Repeat Step 3 for all the other source–destination pairs.
5. For each source–destination pair (S, D), normalize all entries in $PPV_{(S,D)}$, with the smallest PPV element over all source–destination pairs equal to 0 and the largest PPV element over all source–destination pairs equal to 100. Let $NPPV_{(S,D)}(L)$ be equal to the normalized value of $PPV_{(S,D)}(L)$.

Stage 2: (Suppose a request arrives to set up a path between S1 and D1 with B amount of bandwidth)

1. Remove links with a residual bandwidth less than the requested bandwidth B.
2. For each network link L, determine its residual bandwidth RB(L), take the reciprocal of RB(L), and normalize $(RB(L))^{-1}$ to the range 0–100, with the smallest $(RB(L))^{-1}$ value equal to 0 and the largest value equal to 100. Let NRB(L) be equal to the normalized value of $(RB(L))^{-1}$.
3. For the source–destination pair (S1, D1), construct a link weight table $LWT_{(S1,D1)}$, using the following equation:

$$LWT_{(S1,D1)}(L) = NPPV_{(S1,D1)}(L) \times (1 - BWP) + NRB(L) \times (BWP) \tag{5.1}$$

   where BWP is the bandwidth proportion parameter.
4. Run Dijsktra's algorithm to compute a link-weight optimized path between (S1, D1).

Note that Stage 1 is executed only once and when there is a topology change. Stage 2 is performed whenever a path setup request arrives at the network. Figures 1 and 2 present Stages 1 and 2 of the algorithm, respectively, in flowchart format.

6. PERFORMANCE EVALUATION

This section describes the set of experiments used to evaluate the performance of DORA and discusses the experiment results. The experiment scenarios are simulated using ns-2 [9]. We examine the performance of DORA with different BWP parameters (0.1, 0.5, and 0.9) against MIRA and SPF. The two performance metrics used in our experiments are the path-setup rejection ration and the percentage of paths requiring reroute upon topology changes. Clearly, a smaller rejection ratio indicates a better overall resource usage, and the number of paths requiring

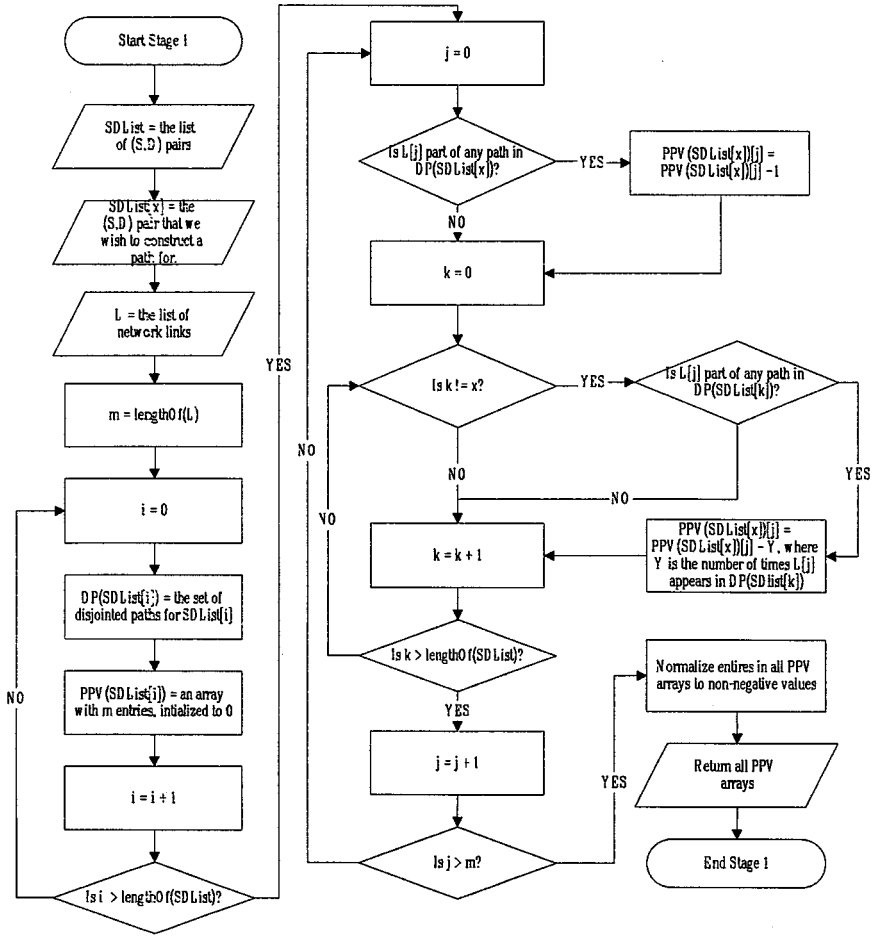

Fig. 1. Operations of Stage 1 of DORA.

reroute should be kept as small as possible to minimize service interruption or delay.

6.1. Simulation Scenarios

Figure 3 shows the network topology chosen for our simulation study. The thicker links have a capacity of 4.8MB while the thinner links have a capacity of 1.2MB. The graph also shows the location of four different source–destination pairs, identified by (S0, D0), (S1, D1), (S2, D2), and (S3, D3).

The size of path setup request is randomly distributed among 10KB, 20KB, 30KB, or 40KB. Four different experiments are carried out. Experiments 1–3

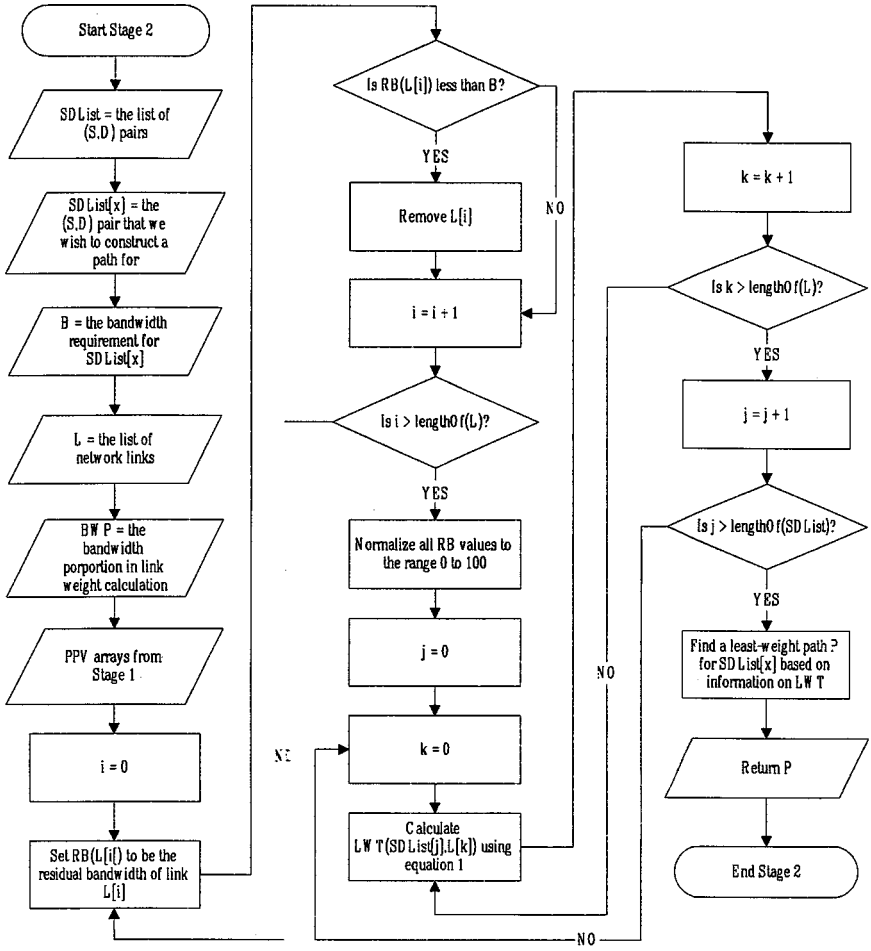

Fig. 2. Operations of Stage 2 of DORA.

examine the path setup rejection ratio. To obtain confidence in the results, each experiment is repeated several times. Experiment 4 investigates the rerouting performance upon topology changes. In Experiment 1, a total of 2000 static path setup requests are sent to the network. Static paths resemble long-lived tunnels in MPLS networks and once established, they will stay in the network forever. In Experiment 2, a total of 2000 dynamic path setup requests are sent to the network. Dynamic paths represent short-lived MPLS tunnels. The arrival of dynamic path setup requests follows a Poisson distribution with mean $\lambda = 80$ requests/time-unit and each dynamic path has a holding time based on an Exponential distribution with mean $\mu = 10$ time-unit. Experiment 3 combines both static and dynamic

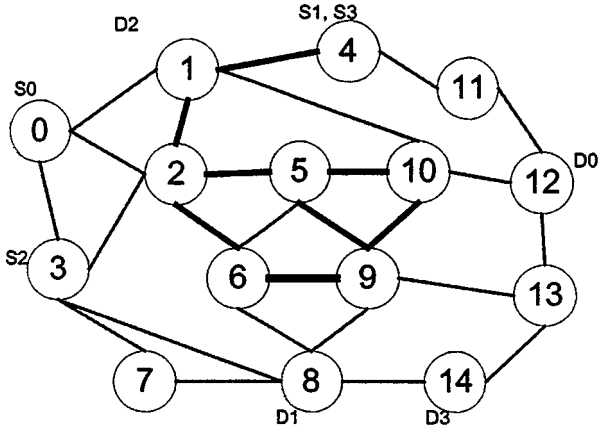

Fig. 3. Network topology used in the simulations.

paths by first loading up the network with 200 static paths, and then sending 1800 dynamic path setup requests to the network. The request arrival rate follows a Poisson distribution with mean $\lambda = 40$ request/time-unit. In Experiment 4, we simulate link failure scenarios by taking down a random link just before network resources are saturated in Experiment 1. The number of paths requiring reroute and the percentage of successful reroutes are recorded after each trial of Experiment 4.

6.2. Path Setup Rejection Ratio

Figure 4 shows the percentage of rejected requests for the static path setup experiment (Experiment 1). Based on the figure, DORA_0.9 (DORA with BWP $= 0.9$) rejects the fewest number of requests, followed by DORA_0.5,

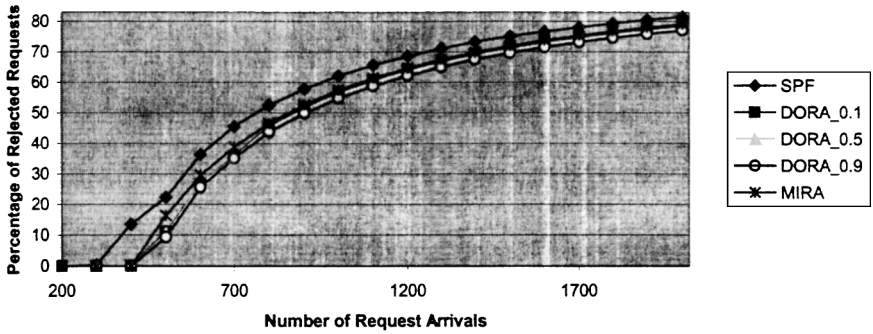

Fig. 4. Static path setup (Experiment 1): Percentage of rejected requests.

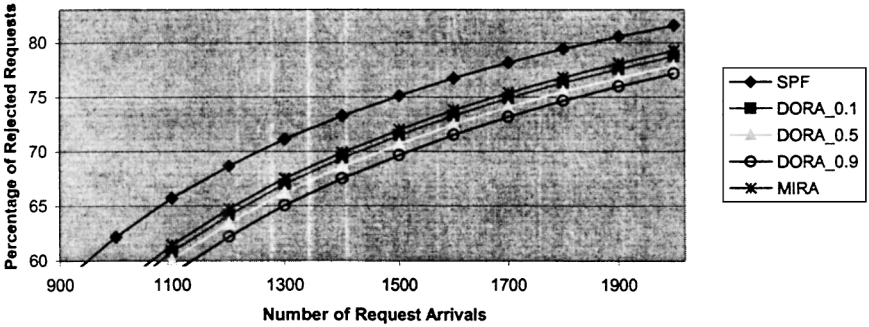

Fig. 5. Static path setup (Experiment 1): Percentage of rejected requests between Request #850 and #2000.

DORA_0.1, MIRA, and finally SPF. The curve for DORA_0.9 stays below all other curves during the course of the simulation, which indicates that it rejects fewer requests than the other algorithms. Since static paths stay in the network forever, after all network resources are saturated, any incoming path setup request will be rejected. This can be observed by the fact that the curves in Fig. 4 approach 100% as the number of request arrivals increases. Figure 5 enlarges the right-most part of Fig. 4, and we can clearly see that DORA_0.9 rejects the fewest number of requests.

The results for the dynamic path setup experiment (Experiment 2) are shown in Fig. 6. In the figure, the curves for DORA stay below the curves for either SPF or MIRA throughout the course of the experiment. All curves grow irregularly until near request #1750, at which all curves enter the steady state and stay flat until the end of the experiment. The three DORA curves overlap and insect each other before entering the steady state. In the steady state, we can see that DORA_0.9 has the lowest rejection percentage, followed by DORA_0.5, DORA_0.1, MIRA, and lastly SPF. DORA_0.9 rejects approximately 21% less requests than MIRA

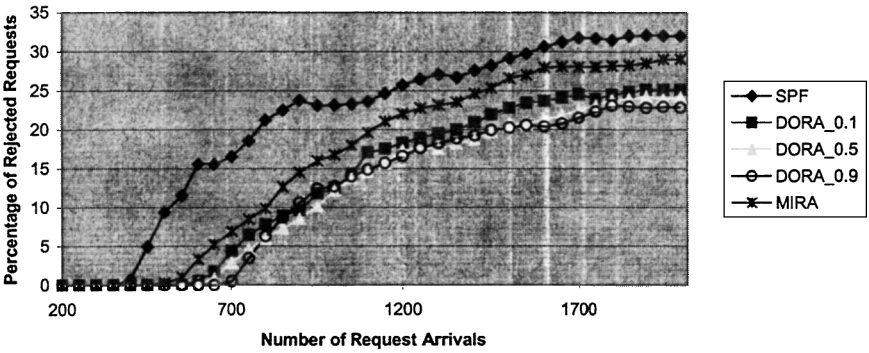

Fig. 6. Dynamic path setup (Experiment 2): Percentage of rejected requests.

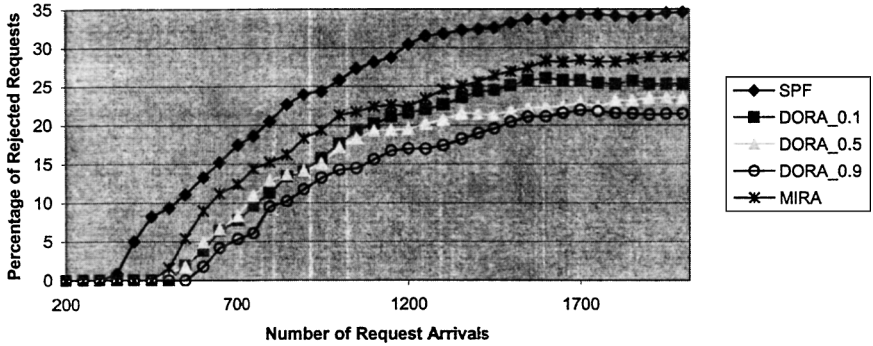

Fig. 7. Static-dynamic path setup (Experiment 3): Percentage of rejected requests.

and 29% less than SPF, and DORA_0.1 shows a 13% improvement on number of rejected requests over MIRA and 22% over SPF.

Experiment 3 involves both static and dynamic path setup requests. As shown in Fig. 7, the curves for DORA grow below SPF and MIRA throughout the course of the experiment. Steady state starts after request #1800 has arrived. Similar to the previous experiments, DORA_0.9 has the lowest rejection percentage, followed by DORA_0.5, DORA_0.1, MIRA, and lastly SPF. In the steady state, DORA_0.9 rejects about 26% less requests than MIRA, and DORA_0.1 shows 12% improvement on number of rejected requests over MIRA. The improvement over SPF is even more significant as DORA_0.9 and DORA_0.1 reject around 37 and 27% less requests than SPF, respectively.

6.3. Path Reroutes and Percentage of Successful Reroutes upon Topology Change

In each trial of Experiment 4, we intentionally take down a link just before network resources are saturated in the static path setup experiment (Experiment 1) and record down the number of paths requiring reroute and the percentage of successful reroutes. There are a total of 26 links as shown in Fig. 3, and we take down each one of them one at a time. In Experiment 1, all incoming requests are rejected when network resources are saturated and it occurs at the point when just above 30% of the total network capacity has been used up. We defined point A, B, and C to be the case where 20, 25, and 30% of the total network capacity has been exhausted. Since there are 26 links in the topology, therefore we have a total of 390 trials (i.e. $26 \times 3 \times 5$) in this experiment. At each link failure point (A, B, and C), we compute the average number of paths requiring reroute, the standard deviation for the number of paths requiring reroute, and the percentage of successful reroutes. The results are shown in Figures 8–10.

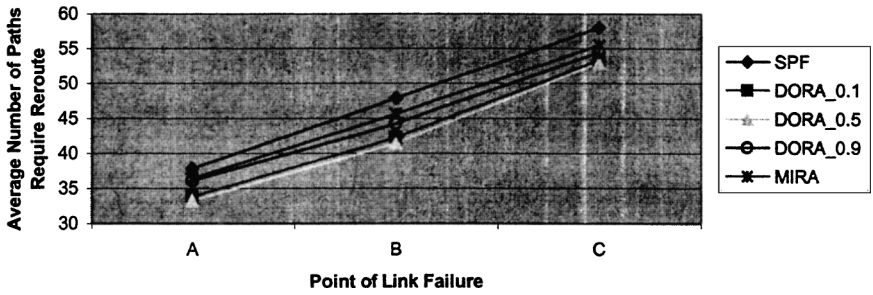

Fig. 8. Average number of paths requiring reroute.

As shown in Fig. 8, the average number of paths requiring reroutes increases as the network is closer to saturation. DORA_0.5 requires the least number of paths to be rerouted upon a link failure, followed by DORA_0.1, DORA_0.9, MIRA, and lastly SPF. The standard deviation value for the number of paths requiring reroute indicates the algorithm's ability to spread path setups evenly across the network. According to Fig. 9, DORA_0.5 has the lowest standard deviation value, meaning that it is the most capable of spreading path setups across the network. In addition, both DORA_0.1 and DORA_0.9 have lower standard deviation values than either MIRA and SPF at all link failure points. Figure 10 shows the percentage of successful reroutes upon link failure. According to the figure, the curves for all algorithms decline as the amount of network resources used increases. DORA_0.5 again performs the best among all algorithms with the highest successful reroutes percentage, followed by DORA_0.9, DORA_0.1, MIRA, and lastly SPF. DORA_0.5 is able to obtain about 2, 7.9, and 6.5% more successful reroutes than MIRA at link failure points A, B, and C respectively. The improvement over SPF is more significant, as DORA_0.5 is able to obtain 18.28, 25.53, and 37.4% more successful reroutes at link failure points A, B, and C respectively. The results for Experiment 4 suggest that a good mix of PPV and residual bandwidth utilization yield the best performance in situations where link failure is commonplace.

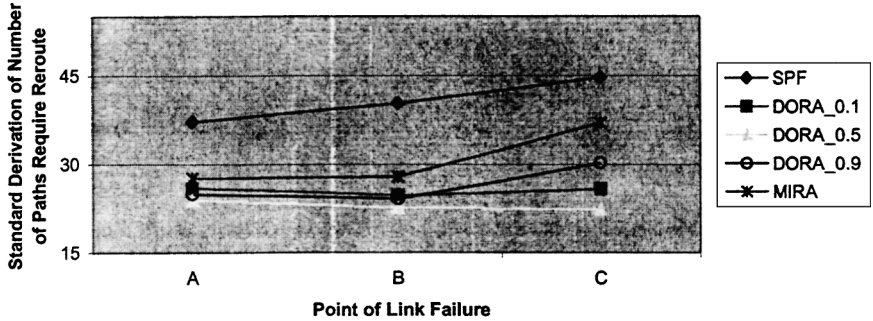

Fig. 9. Standard deviation of number of paths requiring reroute.

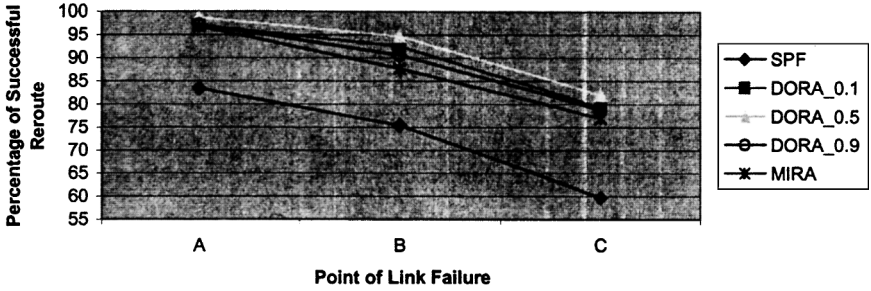

Fig. 10. Percentage of successful reroutes.

6.4. Computation Complexity

Next we will examine and compare the computation complexity of the routing algorithms used in the experiment. Consider a network with N nodes and M links. Let D be the largest degree of any node and P the number of source–destination pairs. Table I shows the time complexity of SPF, MIRA, and DORA. SPF is the least expensive in terms of runtime complexity, but it offers a much worse performance than other algorithms as shown in the previous experiments. The second stage of DORA is equally inexpensive as SPF and is executed during each request arrival. The first stage of DORA is performed only upon network topology change. In the worst case scenario where $D \in O(M)$ and $P \in O(N^2)$, the runtime for Stage 1 of DORA is several magnitudes higher than that of SPF, but still lower than that of MIRA. The dominating computation of MIRA is to determine the maximum flow between each source–destination pair. At least one maximum flow calculation has to be carried out for each source–destination pair, and each calculation takes $O(NM \log(N^2/M))$ time [10].

6.5. Complexity Analysis for DORA

Let N be the number of network nodes, M be the number of network links, and P be the number of edge nodes ($P \leq M$). We will refer to the first step of Stage 1 as Step 1.1 and the second step of Stage 1 as Step 1.2 and so on. For

Table I. Time Complexity for Different Routing Algorithms

Algorithm	Time complexity
SPF	$O(M + N \log N)$
MIRA	$O(N^3 M \log(N^2/M))$
DORA—Stage 1	$O(N^2 M^2 + N^3 M \log N)$
DORA—Stage 2	$O(M + N \log N)$

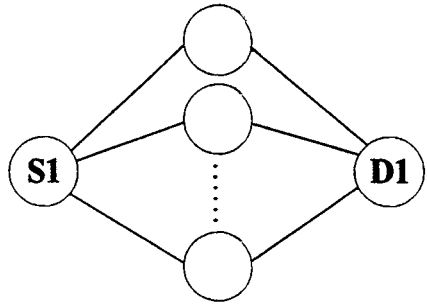

Fig. 11. The number of disjointed paths between a source–destination pair could be as many as $M/2$.

Step 1.1, if every edge node acts as both a source and a destination node, then there will be a maximum of P^2 pairs of source–destination nodes. Computing the set of disjointed paths for each source–destination pair using Dijsktra's shortest path algorithm requires $O(NM)$. This could be improved to $O(M + N \log N)$ by using a priority queue wiht Fibonacci heap in the implementation [11]. Figure 11 shows that the number of disjointed paths for a single source–destination pair could be as many as $M/2 = O(M)$. Hence the worst case runtime for Step 1.1 is $O(P^2NM^2)$. Step 1.2 takes $O(P^2)$ time for PPV array initialization. Steps 1.3 and 1.4 loop through each source–destination pair to compute the PPV array, and both steps combined require $O(P^2M)$ time. Step 1.5 normalizes the PPV array entries to become nonnegative values and it requires $O(P^2)$ time. Hence the runtime for Stage 1 of DORA is dominated by the first step and it is $O(P^2M(M + N \log N))$. In the extreme case, if all the network nodes are also edge nodes, that is $P = N$, then the runtime becomes $O(N^2M^2 + N^3M \log N)$. For Stage 2, each of Steps 2.1, 2.2, and 2.3 requires $O(M)$ time to execute. Step 2.4 computes the least cost path using Dijsktra's algorithm and it requires $O(M + N \log N)$ time. Overall Stage 2 has a time complexity of $O(M + N \log N)$.

7. CONCLUSION AND FUTURE WORK

In this paper, we have introduced DORA for computing bandwidth guaranteed paths in MPLS networks. It combines the PPV and current bandwidth utilization to produce link weights, which in turn are used to find a weight-optimized path. We have shown that DORA rejects fewer path setup requests than both SPF and MIRA. We have also shown that during link failures, DORA requires fewer paths to be rerouted and obtains higher successful reroute percentage than either SPF or MIRA. In addition, DORA is computationally less expensive than MIRA, and has a runtime equal to that of SPF if topology change is infrequent. If node/link failure

occurs, then topology changes will trigger the operation of Stage 1 in DORA. The cost of Stage 1 execution could be reduced by using a better mechanism for determining the set of disjointed paths associated with each source–destination pair. Instead of recalculating the set of disjointed paths every time a topology change occurs, one could recompute only the affected disjointed paths.

One possible extension to DORA is to use past knowledge to estimate the future demand size for each source–destination pair. For instance, instead of incrementing and decrementing the PPV of a link by 1, we may modify the PPV by a value higher than 1 to reflect a larger expected demand size for a given source–destination pair. Such knowledge could be inferred from constant network monitoring and measurements, or derived from the service level agreement between the customer and the service provider.

ACKNOWLEDGMENTS

The authors thank the Canadian Institute for Telecommunications Research (CITR) and the Natural Science and Engineering Research Council of Canada (NSERC) for supporting this project.

REFERENCES

1. K. Kar, M. Kodialam, and T. V. Lakshman, Minimum interference routing of bandwidth guaranteed tunnels with MPLS traffic engineering applications, *IEEE Journal on Selected Areas in Communications: Quality of Service in the Internet*, Vol. 18, No. 12, pp. 921–940, Dec. 2000.
2. S. Suri, M. Waldvogel, and P. R. Warkhede, Profile-based routing: A new framework for MPLS traffic engineering. In *Quality of Future Internet Services*, Lecture Notes in Computer Science, Springer, New York, 2001.
3. B. Fortz and M. Thorup, Internet traffic engineering by optimizing OSPF weights. In *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communication Societies (IEEE INFOCOM)*, Tel Aviv, Israel, March 2000.
4. Y. Wang and Z. Wang, Explicit routing algorithms for internet traffic engineering. In *Proceedings of the 8th Computer Communications and Networks*, Boston, Massachusetts, Oct. 1999.
5. Z. Liu, Y. Sun, and X. Xue, A static routing algorithm used in the Internet traffic engineering. In *Proceedings of the 7th Asian-Pacific Conference on Circuits and Systems (IEEE APCCAS)*, Tianjin, China, Dec. 2000.
6. M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.
7. D. Katz, D. Yeung, and K. Kompella, Traffic engineering extensions to OSPF. Work in progress, Internet Draft, draft-katz-yeung-ospf-traffic-05.txt, 2001.
8. T. Li and H. Smit, IS-IS extensions for traffic engineering. Work in progress, Internet Draft, draft-ietf-isis-traffic-04.txt, 2001.
9. S. McCanne and S. Floyd, Network Simulator 2, http://www.isi.edu/nsnam/ns/.
10. A. V. Goldberg and R. E. Tarjan, A new Approach to the maximum flow problem. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, *Journal of ACM (JACM)*, Vol. 35, No. 4, Oct. 1988.
11. H. R. Lewis and L. Denenberg, *Data Structures and Their Algorithms*, HarperCollins, New York, 1991.

12. E. Rosen, A. Viswanathan, and R. Callon, Multiprotocol label switching architecture, RFC3031, Jan. 2001.
13. D. Awudche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, Requirements for traffic engineering over MPLS, RFC 2702, Sept. 1999.
14. X. Xiao, A. Hanna, B. Bailey, and L. M. Ni, Traffic engineering with MPLS in the Internet, *IEEE Network Magazine*, Vol. 14, No. 2, 2000.
15. R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, New Jersey, 1993.
16. U. Black, *MPLS and Label Switching Networks*, Prentice Hall, New Jersey, 2001.
17. G. Ahn and W. Chun, Design and implementation of MPLS network simulator supporting LDP and CR-LDP. In *Proceedings of the 8th International Conference on Networks (IEEE ICON 2000)*, Singapore, 2000.
18. P. Aukia, M. Kodialam, P. V. N. Koppol, T. V. Lakshman, H. Sarin, and B. Suter, RATES: A server for MPLS traffic engineering, *IEEE Network Magazine*, Vol. 14, No. 2, pp. 34–41, March-April 2000.

Raouf Boutaba has been a Professor at the University of Waterloo in the Department of Computer Science since August 1999. Before that he was a Professor in the Electrical and Computer Engineering Department of the University of Toronto. He has also been an adjunct Professor at the University of Montreal since 1995. Dr Boutaba conducts research in integrated network and systems management, wired and wireless multimedia networks, and quality of service control in Internet networks.

Wayne Szeto received the B Math degree in computer science from University of Waterloo in 2000, and is currently working toward M Math degree. His research interests include constraint-based routing and topology management for virtual overlay networks.

Youssef Iraqi received his MSc degree in computer science from the University of Montreal in 2000. He is now a PhD candidate at the University of Montreal. From 1996 to 1998 he was a research assistant at the Computer Science Research Institute of Montreal. His research interests include QoS control in the Internet, and resource management in wireless mobile networks.