Dynamic Online Routing Algorithm for MPLS Traffic Engineering

W. Szeto, R. Boutaba, and Y. Iraqi

Department of Computer Science,
University of Waterloo,
200 University Avenue West,
Waterloo, ON, Canada, N2L 3G1
{wwszeto, rboutaba, iraqi}@bbcr.uwaterloo.ca

Abstract. The main contribution of this paper is a new online routing algorithm, called Dynamic Online Routing Algorithm (DORA), for dynamic setup of bandwidth guaranteed paths in MPLS networks. The goal of DORA is to accept as many network path setup requests as possible by carefully mapping paths with reserved bandwidth evenly across the network. The key operation behind DORA is to avoid routing over links that: 1) have high potential to be part of any other paths, and 2) have less residual bandwidth available. We compare DORA against other existing constraint-based routing algorithms based on two performance metrics: path setup rejection ratio and percentage of successful reroutes. Our result shows that DORA performs better than the other algorithms in both metrics.

Keywords: MPLS, Constraint-based Routing, Traffic Engineering

1 Introduction

We present a new online routing algorithm, called Dynamic Online Routing Algorithm (DORA), for construction of bandwidth guaranteed paths. DORA aims to utilize existing network resources and minimize network congestions by carefully mapping paths with specific bandwidth requirement evenly across the network topology. The main objective of DORA is to allow more path setups to be accepted into the network, and as a result, increased revenue for service providers.

The problem of establishing bandwidth guaranteed paths, as path setup request arrives one-by-one with no advance about future requests, has been studied elsewhere in [KL00,SWW01,FT00]. Our work is inspired by the Minimum Interference Routing Algorithm proposed in [KL00], but performs better in terms of request rejection ratio and rerouting percentage upon link/node failures with much less computation complexity. We will describe our algorithm in the context of MPLS-enabled networks. In a Multi-Protocol Label Switching (MPLS) network, incoming packets are assigned a label at the ingress node. Each packet follows a pre-computed path, which is identified by a set of labels, to reach its

E. Gregori et al. (Eds.): NETWORKING 2002, LNCS 2345, pp. 936–946, 2002.
© Springer-Verlag Berlin Heidelberg 2002

destination node. Constraint-based Routing (CR) extends MPLS path computation by ensuring that the resulting path satisfies a set of constraints. In this paper, we will consider path computation with bandwidth requirement as the constraint.

The organization for the rest of this paper is as follows: Section 2 identifies important design issues for routing algorithms in a MPLS network. In Section 3, we discuss some related works. Section 4 describes the details of DORA algorithm. Section 5 evaluates and compares DORA against other existing routing algorithms through network simulations. Section 6 summarizes the important points in the paper and proposes future work.

2 Design Issues of Constraint-Based Routing Algorithms

In this section, we briefly describe some of the important properties of a useful routing algorithm in the MPLS domain.

1. **Routing Constraint:** Constraints may include delay, jitter, loss ratio, bandwidth, administrative constraints and so on. It has been proven that finding the optimal route subjected to two or more additive and/or multiplicative metrics is NP-complete [GJ79]. In addition, it is generally difficult to obtain accurate values for certain metrics such as delay and jitter. In the rest of this paper, we will focus on routing algorithms that compute paths subjected to bandwidth requirement.

2. **Online Routing:** Offline constraint-based routing requires a demand matrix as input. A demand matrix describes the expected amount of data to be transmitted between a source-destination pair in the network at different times. In contrast, online constraint-based routing does not require a priori knowledge of the size and arrival time of each individual path setup request. This paper focuses on online constraint-based routing, as it is the appropriate approach to solving the dynamic path setup problem in MPLS networks.

3. **Computational Requirement:** Online routing algorithm compute paths as setup requests arrive at the network. In the case where the ingress node is operating in demand-driven mode, the path computation time is added to the overall response time that the user perceives. Therefore it is necessary for path computations to be as fast and as efficient as possible.

4. **Re-routing Performance:** Network topology changes are triggered by events such as link or node failure. When a link fails, all affected paths have to be re-routed to a different location in order to resume normal operations. Since each path is associated with reserved bandwidth, re-routing may fail due to insufficient residual resource. A good routing algorithm should carefully map network paths across the topology so that when a link fails, the chance of successfully rerouting affected paths is as high as possible.

5. **Link State Distribution:** The current standard link state advertisements (LSA) do not contain dynamic link information such as residual bandwidth and residual capacity. There have been work to extend the Interior Gateway Protocols (IGPs) to add dynamic link attributes to LSAs [KYK02,LS01]. We assume that the necessary bandwidth and topology related information used by the routing algorithm is available when needed.

3 Related Works

The most popular and widely used routing algorithm in MPLS networks is the shortest-path first algorithm (SPF) based on the number of hops. SPF selects the path that contains the fewest hops between the source and the destination node. One obvious problem with SPF is that it tends to route traffic onto the same set of links until these links' resource are exhausted. This leads to concentration of traffic on certain parts of the network. In addition, SPF typically accepts less path setups into the network than some other more advanced routing algorithms.

A more intelligent routing algorithm the Minimum Interference Routing Algorithm (MIRA) proposed in [KL00]. The objective of MIRA is to accept as many path setups into the network as possible by using the concept of critical links. Critical links have the property that when their capacity is reduced by 1 bandwidth-unit, the maximum data flow between a given source-destination node is also reduced by 1 bandwidth-unit. Therefore the goal of MIRA is accomplished by selecting paths that contain as few critical links as possible. However MIRA suffers from two weaknesses. First, MIRA is computationally expensive, because it needs to perform a lot of maximum data flow calculations, and each max flow computation is $O(N^3)$ [SWW01], where N is the number of nodes in the network. Compare to the runtime of SPF, which is $O(NM)$ (M is the number of links in the network), the runtime for MIRA is several magnitudes higher. Second, in some situations, MIRA may continuously choose the same set of links to route traffics on. To illustrate this point, consider the situation where there exist two distinct routes with identical residual bandwidth connecting the same source-destination pair (S, D). Initially all links in both route are classified as critical links. When a request associated with (S, D) arrives, given sufficient, one of the two routes will be chosen to serve this request. Afterwards, all the links in the chosen route are no longer critical, but all links in the other route remain critical links. This means subsequent requests will be routed on the same route as the first request until resources along this route are exhausted.

Some other related work includes profile-based routing and variations of OSPF routing heuristics proposed in [FT00] and [SWW01] respectively. These two routing algorithms require a traffic profile or an estimated demand matrix structure that approximates the bandwidth demands between each pair of source-destination nodes at different times. They will not be used for performance comparison purposes because they are not strictly online routing algorithms. Instead, we will compare SPF, MIRA and DORA in the performance evaluation section.

4 Dynamic Online Routing Algorithm

We consider the problem of setting up bandwidth guaranteed paths in a MPLS network. Each path setup request arrives one-by-one and we do not know the arrival time or size of future requests. The size of a request refers to the bandwidth requirement of the path to be setup. Each request demands a path with reserved bandwidth to be setup between an ingress (source) node and an egress (destination) node.

DORA is separated into two stages. Stage one is executed whenever a topology change has occurred, and stage two is executed whenever a path setup request arrives to the network. In the first stage, the key operation is to assign path potential value (PPV) to each link with respect to each source-destination pair. PPV reflects how likely a particular link will be part of some potential paths between some source-destination pairs in the network. A large PPV link value implies that this link will likely be part of many potential paths and thus routing over this link should be avoided whenever possible. A small PPV link value means that there are less potential paths using this link and therefore it is more desirable to use this link than others with larger PPV value. Each source-destination (S, D) is associated with an array, $PPV_{(S,D)}$, of size equal to the number of network links and each array element is initialized to zero. The way that $PPV_{(S,D)}$ array elements are calculated is based on which links are included in the disjointed set of paths for each source-destination pairs and it is described in detail in the pseudo-code listing for DORA. The main idea is to go through each source-destination pair (S, D), reduce one from $PPV_{(S,D)}(L)$ if the link L appears in the disjointed path set for (S, D), and add one to $PPV_{(S,D)}(L)$ each time L appears in any of the disjointed path sets associated to any other source-destination pairs. In the second stage, the PPV for each link is combined with the reciprocal of the current residual bandwidth of this link to form the link weight. The content of the link weight is controlled by a user parameter BWP (bandwidth proportion), which takes on 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 30% of the link weight is contributed by the associated PPV.

Stage 1:

1. For each ingress-egress 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 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 ingress-egress pair (S_1, D_1), construct the $PPV(S_1, D_1)$ array, and initialize all entries to zero. The size of the array is equal to the number of network links.

3. For each ingress-egress pair (S_1, D_1):

   a) Go through each link in the network and if a link L is part of any paths in $DP_{(S_1,D_1)}$, subtract 1 from $PPV_{(S_1,D_1)}(L)$.

   b) For all the ingress-egress pair other than (S_1, D_1), inspect each link L and determine the number of times, X, that L appears in $DP_{(S,D)}$ where (S, D) not equal to (S_1, D_1). Increment $PPV_{(S,D)}(L)$ by X.

4. Repeat step 3 - 4 for all the other ingress-egress pairs.

5. For each ingress-egress pair (S, D), normalize all entries in $PPV_{(S,D)}$, with the smallest PPV element over all ingress-egress pairs equal to 0 and the largest PPV element over all ingress-egress pairs equal to 100. Let $NPPV_{(S,D)}(L)$ to be equal to the normalized value of $PPV_{(S,D)}(L)$.

Stage 2:

1. Suppose a request arrives for path setup between (S_1, D_1) with Y amount of bandwidth. Remove links with residual bandwidth less than the requested bandwidth Y.

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 to 100, with the smallest $RB(L) - 1$ equal to 0 and the largest $RB(L) - 1$ equal to 100. Let $NRB(L)$ to be equal to the normalized value of $RB(L) - 1$.

3. For the ingress-egress pair (S_1, D_1), construct a link weight table $LWT_{(S_1,D_1)}$, and using the following equation to obtain $LWT_{(S_1,D_1)}(L)$:

$$LWT_{(S_1,D_1)}(L) = NPPV_{(S,D)}(L) \times (1 - BWP) + NRB(L) \times BWP \quad (1)$$

4. Run Dijsktra's algorithm to compute a link weight-optimized path between (S_1, D_1).

5 Performance Evaluation

In this section, we will first describe the set of experiments used to evaluate the performance of DORA, and then we will comment on the results of the experiments. All experiment scenarios are simulated using ns-2 [BEF+00]. We compare the performance of DORA with different BWP parameter values (0.1, 0.5, and 0.9) against that of SPF and MIRA. Two main performance metrics of interest are the path setup rejection ratio and the percentage of paths successfully rerouted upon topology change. The network topology used in the experiments

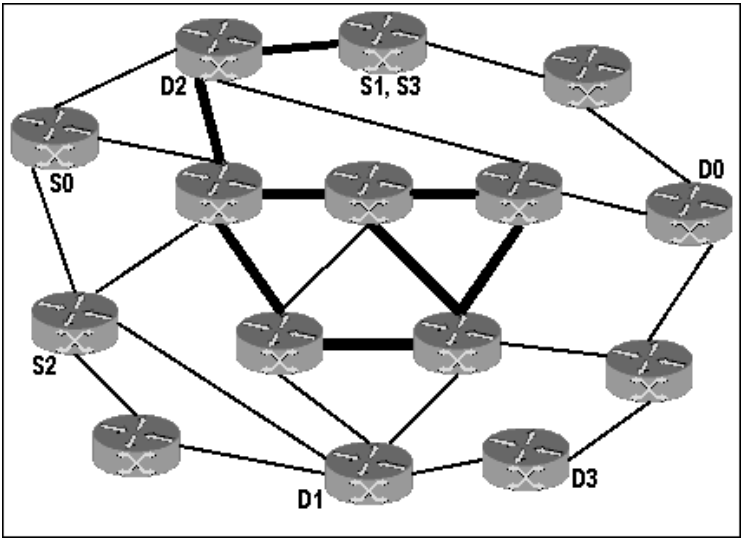

Fig. 1. Network topology used in the experiments. The thicker lines represent links with 48MBytes of reservable bandwidth while the thinner lines represent links with 12MBytes of reservable bandwidth.

represents a small ISP's backbone network and is shown in Figure 1. The figure also shows the location of 4 different source-destination pairs, identified by (S_0, D_0), (S_1, D_1), (S_2, D_2) and (S_3, D_3).

Three different experiment scenarios are considered. The first two experiments focus on the path rejection ratio, which indicates the percentage of path setups that are rejected due to insufficient resources. The last experiment studies the routing algorithm behaviors upon link failures. The size of each path setup request or the bandwidth requirement of the path is uniformly distributed among 10KB, 20KB, 30KB and 40KB. In experiment 1, a total of 2000 static path setup requests are sent to the network. Static paths resemble long-lived MPLS tunnels and once they are established, they will stay in the network forever. In experiment 2, we first load up the network with 200 static paths, and then we send 1800 dynamic path setup requests to the network. Dynamic paths represent short-lived MPLS tunnels. The arrival time of dynamic path setup requests at the network is based on a Poisson distribution with mean λ=40 requests per time-unit and each dynamic path has a holding time based on an Exponential distribution with mean μ=10 time-unit. The setup of experiment 3 is the same as that of experiment 1 with an additional setting - a randomly chosen link is taken down just before network resources are saturated. The number of paths requiring reroute and the percentage of successful reroutes are recorded after a link is taken down.

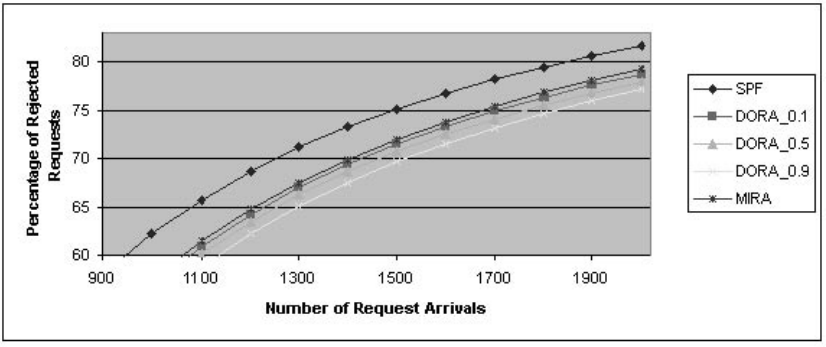

Fig. 2. Static Path Setup (Experiment 1): Percentage of Rejected Requests between Request #850 and #2000.

Figure 2 shows the partial result for experiment 1, which involves only static path setup requests. According to the figure, DORA_0.9 (DORA with $BWP = 0.9$) rejects the fewest number of requests, followed by DORA_0.5, DORA_0.1, MIRA and finally SPF. The result before setup request #850 is similar - that is the relative positions of the curves are the same during the course of the simulation. Since static paths remain in the network forever, after all network resources are exhausted, any incoming path request will be rejected. This can be observed by the fact that all curves in the figure approaches 100% as the number of request arrival increases.

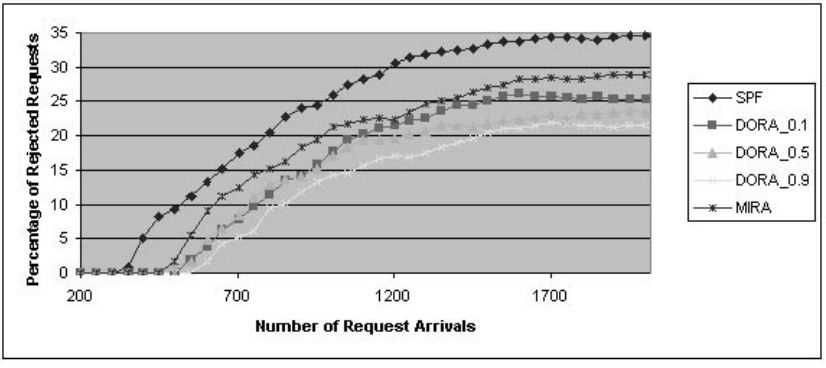

Fig. 3. Static-Dynamic Path Setup (Experiment 2): Percentage of Rejected Requests.

The result for experiment 2, which involves both static and dynamic path setup requests, is shown in Figure 3. In the figure, all curves grow irregularly

until around request #1800, at which all curves enter steady state and stay relatively flat. Similar to the previous experiment, DORA_0.9 rejects the least percentage of requests, followed by DORA_0.5, DORA_0.1, MIRA and lastly SPF. During 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 rejects around 37% and 27% less requests than SPF, respectively.

Experiment 3 is equivalent to experiment 1 except that at different points in time, a randomly chosen link is taken down and the number of paths requiring reroute and the percentage of successful reroutes are recorded. In experiment 1, the condition where all incoming requests are rejected due to insufficient resources occurs when just above 30% of the total network capacity has been occupied. We defined point A, B, and C to be the case where 20%, 25%, 30% of the total network capacity has been saturated. There are a total of 26 network links which yields a total of 390 experiment trials (e.g. $26 \times 3 \times 5$). At each link failure point (A, B, and C), we compute the average number of paths requiring reroute, the standard deviations on the number of paths requiring reroute, and the percentage of successful reroutes. The results are shown in Figure 4, Figure 5, and Figure 6.

Figure 4 shows the average number of paths requiring reroute increases, as the network resource is closer to exhaustion. The least number of paths are required to be rerouted upon a link failure by using DORA_0.5, followed by DORA_0.1, DORA_0.9, MIRA and finally SPF. The standard deviation value for the number of paths requiring reroute is a direct indication of the algorithm's ability to spread path setups evenly across the network.

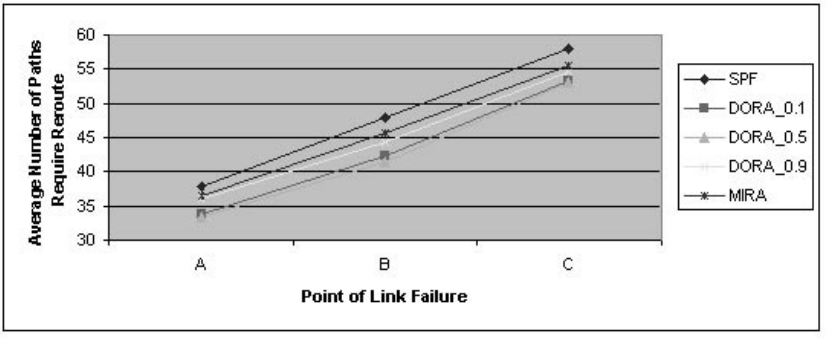

Fig. 4. Average Number of Paths Requiring Reroute at Different Failure Points.

Figure 5 shows that 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 value than either MIRA and SPF at all link failure points.

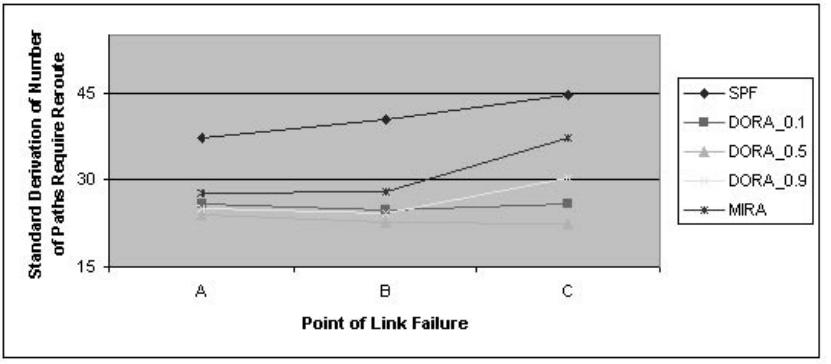

Fig. 5. Standard Deviation of Number of Paths Requiring Reroute at Different Failure Points.

Figure 6 shows the percentage of successful reroutes upon link failure. According to the figure, the curve for all algorithms declines, as the amount of network resource is closer to saturation. 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 path potential value and residual bandwidth utilization yield the best performance in situations where link failure is commonplace.

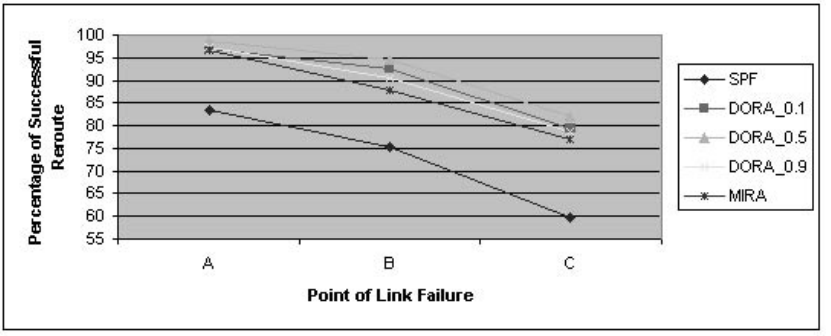

Fig. 6. Percentage of Successful Reroutes at Different Failure Points.

Next we will examine and compare the computation complexity of the routing algorithms used in the experiment. Consider a network consisting of N nodes and M links. Let D be the largest degree of any node and P be the number of source-destination pairs. Table 1 shows the computation complexity between the shortest-path algorithm (SPF), the minimum interference routing algorithm (MIRA), and our algorithm (DORA). SPF is 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 absolute worst case scenario where $D = O(M)$ and $P = O(N^2)$, the runtime for stage one of DORA is several magnitudes higher than that of SPF, but still lower than that of MIRA.

Table 1. A comparison of computation complexity between different routing algorithms.

Algorithm	Computation Complexity
SPF	$O(MN)$
MIRA	$O(N^5) + O(M^2)$
DORA Stage 1	$O(N^3M^2)$
DORA Stage 2	$O(MN)$

6 Conclusion

In this paper, we have introduced Dynamic Online Routing Algorithm (DORA) for computing bandwidth guaranteed paths in MPLS networks. It combines the path potential value and current residual bandwidth to construct the link weight table with respect to each source-destination pair. A weight-optimized path based on the associated link weight table is then computed and returned by DORA. In the performance evaluation section, we have shown that DORA rejects fewer path setup requests than both SPF and MIRA. Furthermore, DORA attains a higher successful reroutes percentage upon link failures than both SPF and MIRA. In addition, the runtime complexity of DORA is less than that of MIRA, and it has an equivalent computation complexity to SPF when topology change is infrequent. When topology change does occur, it will trigger the execution of stage 1 in DORA. The cost of stage 1 operation could be reduced by using a better scheme for computing the set of disjointed paths with respect to each source-destination pair. Additionally, instead of recalculating the set of disjointed paths every time a topology change occurs, 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 in-

crementing and decrementing the PPV of a link by one, we may modify the PPV by a value higher than one 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.

References

[AC00] G. Ahn and W. Chun. Design and implementation of mpls network simulator supporting ldp and cr-ldp. In *Proceedings of ICON*, 2000.

[AMA+99] D. Awudche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. Requirements for traffic enginering over mpls. *RFC2702*, 1999.

[AMO93] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[BEF+00] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu. Advances in network simulation. *IEEE Computer Magazine*, 33(5):59–67, May 2000.

[Bla01] U. Black. *MPLS and Label Switching Networks*. Prentice Hall, 2001.

[FT00] B. Fortz and Mikkel Thorup. Internet traffic engineering by optimizing ospf weights. In *Proceedings of INFOCOM*, 2000.

[GJ79] M. Garey and D. Johnson. *Computer and Intractability*. W. H. Freeman, 1979.

[KL00] M. Kodialam and T.V. Lakshman. Minimum interference routing with applications to mpls traffic engineering. In *Proceedings of INFOCOM*, 2000.

[KYK02] D. Katz, D. Yeung, and K. Kompella. Traffic engineering extensions to ospf. *Internet Draft draft-katz-yeung-ospf-traffic-06.txt*, 2002.

[LS01] T. Li and H. Smit. Is-is extensions for traffic engineering. *Internet Draft draft-ietf-isis-traffic-04.txt*, 2001.

[RVC01] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. *RFC3031*, 2001.

[SWW01] 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 2156*. Springer Verlag, Sept 2001.

[XHBN00] X. Xiao, A. Hanna, B. Bailey, and L. M. Ni. Traffic engineering with mpls in the internet. *IEEE Network Magazine*, pages 28–33, March 2000.